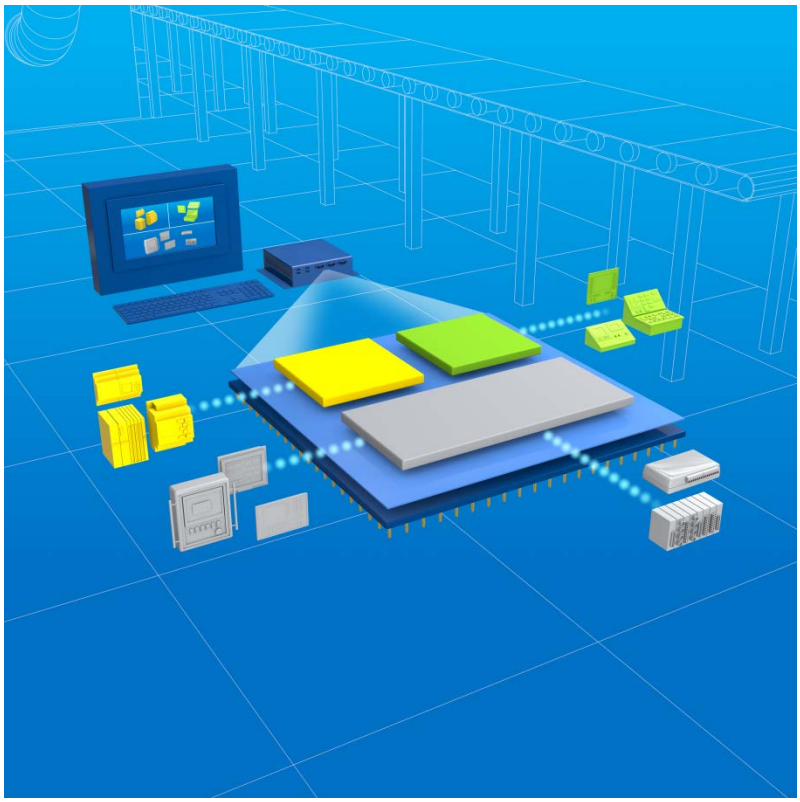
Intelligent Systems

Intel® Industrial Solutions System Consolidation Series User's Guide

For SCS Version 1.0

June 20, 2014

Revision 1.1



Document #538056-1.1

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel® Industrial Solutions System Consolidation Series and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands (denoted by an asterisk upon first instance in this document) may be claimed as the property of others.

Copyright © 2014, Intel Corporation. All rights reserved.

Contents

1 Introduction 12

1.1 Terminology 13
1.2 Product Literature 13
1.3 Reference Content 14

2 Safety Notice 15

2.1 Alerts for Warning, Caution, Important, and Note 15
2.1.1 WARNING 15
2.1.2 CAUTION 15
2.1.3 IMPORTANT 15
2.1.4 NOTE 15
2.2 Safety During Installation and/or Maintenance 16
2.3 Safety Related to Data Preservation 16

3 Intel® Industrial Solutions System Consolidation Series Documentation 17

4 User-Provided Items 19

4.1 Required User-Provided Items 19
4.1.1 Development Host 19
4.1.2 Development Host Specifications 19
4.2 Optional User-Provided Items 20

5 Development Workflows 21

5.1 Summary of Workflows 21

6 Workflow 1: Set Up & Explore the System Features 22

6.1 About This Chapter 22
6.2 System Introduction 23
6.3 User-Provided System Components 24
6.4 Target Platform Subsystem Overview 25
6.4.1 Virtual NIC (VNIC) 27
6.4.2 Default Target OS IP Address Assignment 28
6.4.3 Customizable Target OS IP Address Assignment 28
6.4.4 Subsystem Connectivity 28
6.4.5 Linux and VxWorks Operating Systems 28
6.5 Development Host Subsystem Overview 29
6.5.1 Development Host Linux Shell 31
6.5.2 Wind River Workbench 32
6.5.3 Special Version of Workbench 32
6.5.4 Workbench Projects 33
6.5.5 The Target Image 34
6.5.6 Development Host Hardware 35
6.5.7 Development Host Installation Software 35
6.6 License the System 35
6.6.1 Target Platform Software Licensing 35
6.6.2 Development Host Software Licensing 36
6.6.2.1 Determine the Host-ID / Host Name 36
6.6.2.2 Locate the Serial Number 38

6.6.2.3 Go To License Website 38
6.6.2.4 Install the License 39
6.7 Install the Development Host Software 40
6.8 Set Up the Target Platform & Development Hosts 50
6.8.1 Set Up the Target Platform 50
6.8.2 Connect the Target Platform and Development Hosts 51
6.9 Explore the System 54
6.9.1 Default OS IP Address Assignment 58
6.9.2 Inter-OS Communication over the Target VNIC 58
6.9.2.1 VxWorks-to-Linux Communication 59
6.9.2.2 Linux-to-VxWorks Communication 59
6.9.2.3 VxWorks-to-VxWorks Communication 61

7 Workflow 2: Develop With Preloaded Workbench Projects 63
7.1 About This Chapter 63
7.2 About Build/Rebuild 63
7.3 Start Up 64
7.4 Workbench Projects 65
7.5 Modify Code in a Preconfigured OS Project 67
7.5.1 Modify Simple Code Line in VxWorks 1 Project 67
7.5.2 Build the Hypervisor Integration Project 70
7.5.3 Copy the SYSTEM.ELF File to Boot Media 73
7.6 Configure Target Platform BIOS to Boot from USB Media 75

8 Workflow 3: Build Linux & VxWorks OS Images from Scratch 79
8.1 About this Workflow 79
8.2 Preface 79
8.3 The VxWorks Source Build (VSB) Project 80
8.4 Create VxWorks Guest OS #1 80
8.4.1 Create the VxWorks Image Project (VIP) 80
8.4.2 Add Components to VxWorks 84
8.4.3 Add User Applications to the VIP Using DKM 90
8.4.4 Add Source Code to the Project 94
8.4.5 Build the Project 97
8.4.6 Build the VIP 98
8.5 Create VxWorks Guest OS #2 100
8.5.1 Create the second VxWorks Image Project (VIP) 100
8.5.2 Add Components to VxWorks 103
8.5.2.1 Exclude DISK_UTIL 106
8.5.3 Add User Applications to the VIP Using Real-Time Process (RTP) 106
8.5.4 Create a ROMFS File System Project 109
8.5.5 Build the Project 113
8.5.6 Build the VIP 114
8.6 Create and Build WR Linux 115
8.6.1 Create a WR Linux Platform Project 115
8.6.2 Configure and Build the File System and Packages 121
8.6.3 Add a User Application to Linux 125
8.7 Build the Hypervisor Integration Project (HIP) 128
8.7.1 Modifying the HIP 128
8.7.2 Build the Hypervisor Project 129
8.8 Copy SYSTEM.ELF to Target Platform USB Media 130
8.9 Open a Development Host AMIO Connection 131
8.10 Configure Target Platform BIOS to Boot from USB Media 132

8.11 Confirm Sample Code Functionality 134
8.11.1 Verify DKM Functionality In VxWorks 1 OS 134
8.11.2 Verify RTP Functionality In VxWorks 2 OS 135

9 Workflow 4: Dynamically Load & Debug Applications on the Target Platform 137
9.1 About This Chapter 137
9.2 Initialize Linux User Mode Agent 137
9.3 Create a Linux User Application 142
9.4 Download Application to a Linux Platform OS 143
9.5 Debug Linux Target Platform Content 147
9.6 Creating VxWorks Target Connection 155
9.6.1 Setting Up Linux Proxy Agent 155
9.6.2 Creating VxWorks Debug Connection 157
9.7 Create a Sample VxWorks Application 161
9.8 Download Application to a VxWorks Platform OS 161
9.9 Debug VxWorks Target Platform Content 164

10 Workflow 5: Assign Persistent Target Platform OS IP Addresses 169
10.1 About This Chapter 169
10.2 Default OS IP Address assignment 169
10.2.1 Customizable Target OS IP Address Assignment 170
10.2.2 Customizable Target OS IP Address Assignment 170
10.2.2.1 Configure a Persistent VxWorks IP Address – 1st Instance170
10.2.2.2 Configure a Persistent VxWorks IP Address – 2nd Instance172
10.2.3 Rebuild Hypervisor and Boot Target Platform 172
10.2.4 Boot the Target Platform with Persistent IP Address Changes 172
10.3 Configure Target Platform BIOS to Boot from USB Media 175
10.3.1 VxWorks-to-Linux Communication 178
10.3.2 Linux-to-VxWorks Communication 179
10.3.3 VxWorks-to-VxWorks Communication 180

11 Workflow 6: Configure a New Device 181
11.1 About This Chapter 181
11.1.1 Special Version of Hypervisor 181
11.1.2 Hypervisor Resource Assignment 182
11.1.3 User-Configurable Resources 183
11.1.4 Default Physical Device Allocations 184
11.1.5 Modify User-Configurable Resources 185
11.1.6 Assign Resources to a Target OS 185
11.1.6.1 OVERRIDE.ESH Script File 187
11.1.6.2 Device Assignment Parameters 188
11.2 Configure Target Platform BIOS to Boot from USB Media 194

Appendix A – Default I/O Device Assignment 199
A.1 Default Physical Device Allocations 199

Appendix B – Technical Support 201
B.1.1 In-Product Technical Support 201
B.2 Online & Live Technical Support 201
B.3 Wind River Product Support & Training 201

Appendix C – Troubleshooting 202
C.1 Installation Troubleshooting 203

C.2 Configuration Troubleshooting 205

Appendix D – Legal Notices 207

D.1 All Notices 207

D.2 License Notices 207

D.3 GNU General Public License 208

Figures

Figure 1 – SCS Target Platform Box Shipment 23
Figure 2 – Development Host and Target Platform System 24
Figure 3 – The SCS Target Platform Hardware 25
Figure 4 – Target Platform OS Configuration 26
Figure 5 – Target Platform IO Configuration 27
Figure 6 - Target Boot Image Project Compilation 30
Figure 7 - Development Host Desktop 31
Figure 8 - Workbench Workspace 32
Figure 9 - Project Explorer 33
Figure 10 – Modifiable Projects 34
Figure 11 – Unmodifiable Projects 34
Figure 12 – Rename the Licensing File 40
Figure 13 – Linux Desktop 41
Figure 14 – Install Icon 41
Figure 15 – Warning Window 42
Figure 16 – Examining Devices 42
Figure 17 – Location Window 43
Figure 18 – Installation Type 44
Figure 19 – Storage Space 45
Figure 20 – Install Target Devices 46
Figure 21 – Examining Devices 46
Figure 22 – System Installs 47
Figure 23 – Installation Complete 48
Figure 24 – Linux Desktop 49
Figure 25 - Power Inverter Plug 50
Figure 26 - Target Platform Serial Connector 51
Figure 27 - Development Host Serial Connector 52
Figure 28 – Null Modem/Gender Changer 53
Figure 29 - Development Host Desktop 54
Figure 30 - Workbench Workspace 55
Figure 31 - AMIO Console 55
Figure 32 - Multiplexed I/O Button 56
Figure 33 – Open AMIO Console 56
Figure 34 - Disconnect Icon 57
Figure 35 – Terminated Connection 57
Figure 36 - AMIO Consoles 58
Figure 37 - VxWorks to Linux Communication 59
Figure 38 – Linux Prompt 60
Figure 39 - Linux-to-VxWorks Communication 60
Figure 40 - Ifconfig Output 61
Figure 41 - VxWorks-to-VxWorks Communication 62
Figure 42- Linux Desktop 64
Figure 43 - Workbench Icon 64
Figure 44 - Workbench Workspace 65
Figure 45 - Project Explorer 66
Figure 46 - VxWorks Project Icon 67
Figure 47 - usrAppInit.c 68
Figure 48 – Print File Code 69
Figure 49 – File > Save 69
Figure 50 – Rebuild Project 70
Figure 51 – Rebuild Project 71
Figure 52 – Build Console 71

Figure 53 – Select Rebuild Project 72
Figure 54 - Build Console 73
Figure 55 – System.elf Copy 74
Figure 56 – System.elf Directory 75
Figure 57 - USB 3.0 Ports 76
Figure 58 – BIOS 1 76
Figure 59 – BIOS 2 77
Figure 60 – BIOS 3 77
Figure 61 – AMIO Consoles 78
Figure 62 - File > New > VxWorks Image Project 81
Figure 63 - Create Project in Workspace 82
Figure 64 - vsb_wrVbX86 83
Figure 65 - Project Details 84
Figure 66 - Components Tab 85
Figure 67 - Find Window 86
Figure 68 - Find Window 2 88
Figure 69 - Highlighted Item 89
Figure 70 - Bolded Item 89
Figure 71 – New DKM 90
Figure 72 - Create... 91
Figure 73 - Build Specs Window 92
Figure 74 – COREgnu 93
Figure 75 - DKM Module 94
Figure 76 – File System 95
Figure 77 – File System 96
Figure 78 – Project Reference 97
Figure 79 – Kernel Object 98
Figure 80 – Build Project 99
Figure 81 – Build Console 99
Figure 82 – VxWorks Image Project 100
Figure 83 – Create Project... 101
Figure 84 – Based On 102
Figure 85 – Expanded Project 103
Figure 86 - Components 104
Figure 87 - Name 105
Figure 88 – VxWorks RTP 107
Figure 89 – Set Active... 108
Figure 90 - COREgnu_RTP 109
Figure 91 - my_romfs 110
Figure 92 - New VxWorks ROMFS 111
Figure 93 – my_romfs Tab 112
Figure 94 - helloworld.exe 113
Figure 95 - ROMFS Project 114
Figure 96 – Build Project 115
Figure 97 – Linux Project 116
Figure 98 - Project 116
Figure 99 – Configure Options 1 117
Figure 100 – Configure Options 2 119
Figure 101 - Processing 120
Figure 102 – Build Target 121
Figure 103 - Add 122
Figure 104 – File Content 123
Figure 105 - Find 124
Figure 106 – y/n/m 125
Figure 107 - Wind River Application Project 125

Figure 108 – General Icon 126
Figure 109 - Import 127
Figure 110 – Hypervisor Updating 128
Figure 111 – Project Name 129
Figure 112 – Open Icon 131
Figure 113 - Channel 132
Figure 114 – BIOS 1 133
Figure 115 – BIOS 2 133
Figure 116 – BIOS 3 134
Figure 117 - VxWorks 1 Code 135
Figure 118 - VxWorks 2 Code 136
Figure 119 - Usermode Agent Output 138
Figure 120 - Agent Monitor Port 138
Figure 121 – Remote Systems Window 139
Figure 122 – White Space 139
Figure 123 – System Type 140
Figure 124 – Target Server… 141
Figure 125 – Remote Connection 142
Figure 126 – Download Configuration 144
Figure 127 – Edit File Copy/Deploy 145
Figure 128 – Edit Window 146
Figure 129 – AMIO Console 147
Figure 130 – Debug Menu Item 148
Figure 131 – Choose Action Window 149
Figure 132 – Create, Manage… 150
Figure 133 – Edit Window 151
Figure 134 – Debug and Console 152
Figure 135 – Debug Menu Items 153
Figure 136 – “Hello World” 154
Figure 137 – Code Content 154
Figure 138 - Usermode Agent Output 156
Figure 139 - Agent Monitor Port 156
Figure 140 – Remote Systems Window 157
Figure 141 – White Space 157
Figure 142 – System Type 158
Figure 143 – Backend 159
Figure 144 – Remote Connection 160
Figure 145 - VxWorks Live Tasks 160
Figure 146 – Download Configuration 162
Figure 147 - Downloads Tab 163
Figure 148 – Successful DKM Download 164
Figure 149 – Debug Menu Item 165
Figure 150 – Choose Action Window 166
Figure 151 – Debug Configurations 167
Figure 152 – Entry Point Selection 168
Figure 153 – usrAppInit.c IP Change 1 170
Figure 154 - usrAppInit.c IP Change 2 171
Figure 155 - Rebuild VxWorks Project 171
Figure 156 – System.elf Copy 173
Figure 157 – System.elf Directory 174
Figure 158 - USB 3.0 Ports 175
Figure 159 – BIOS 1 176
Figure 160 – BIOS 2 176
Figure 161 – BIOS 3 177
Figure 162 –AMIO Consoles 177

Figure 163 - Ping Command 178
Figure 164 - Ping Command 179
Figure 165 - Ping Command 180
Figure 166 – Target Platform Devices 1 184
Figure 167 - Target Platform Devices 2 185
Figure 168 - Hypervisor Project 186
Figure 169 - OVERRIDE.ESH Work Pane 187
Figure 170 - OVERRIDE.ESH Configurable Section 187
Figure 171 - Rebuild Hypervisor Project 191
Figure 172 – System.elf Copy 192
Figure 173 – System.elf Directory 193
Figure 174 - USB 3.0 Ports 194
Figure 175 – BIOS 1 195
Figure 176 – BIOS 2 195
Figure 177 – BIOS 3 196
Figure 178 – Hypervisor AMIO Console 196
Figure 179 – Device Information 197
Figure 180 – Device BDF and [VendorID]:[DeviceID] Information 198
Figure 181 – Target Platform Devices 1 199
Figure 182 - Target Platform Devices 2 200

Tables

Table 1 – Terminology 13
Table 2 – Product Literature 13
Table 3 – Reference Documents 14
Table 4 – VIP INLCUDE Components 87
Table 5 - Device Classifications 182
Table 6 - User- Configurable Resources 183
Table 7 - Supported Guest OSes 188
Table 8 - Sample Scenario 189
Table 9 - Installation Troubleshooting 203
Table 10 - Installation Troubleshooting 205

Revision History

Date	Revision	Description
2/12/2014	1.0	Official release for Intel® Industrial Solutions System Consolidation Series (SCS) version 1.0
6/20/2014	1.1	New licensing instructions added

§

1 Introduction

This document is written for use by system developers including embedded system developers. This content assumes advanced knowledge of installing and configuring hardware and software for a personal computer system. Engineers using this document should be familiar with the use of and programming of:

- Wind River* Hypervisor
- Wind River* Linux*
- Wind River* VxWorks*
- Wind River* Workbench

IMPORTANT: Read and understand this document in its entirety before performing any activities such as installing software and starting up the system.

IMPORTANT: This content may be updated without notice. To ensure that you are reading the most recent document, check the product support website:

www.intel.com/industrialconsolidation

Upon downloading a copy of the *User Guide* from the website, compare the cover page date of the downloaded document to the *User Guide* you currently have.

1.1 Terminology

Table 1 – Terminology

Term	Description
Development host	The portion of the SCS system used for development. Configurations created on the *development host* are loaded from the *development host* onto the *target platform* to realize productivity.
Intel® Industrial Solutions System Consolidation Series	*Intel® Industrial Solutions System Consolidation Series* is also sometimes known by its shorter name, SCS.
PC	Personal computer
SCS	Intel® Industrial Solutions **System Consolidation Series**
System	The Intel® Industrial Solutions System Consolidation Series *system* comprises two subsystems: • Target platform • Development host with Software Features For detailed descriptions of these items, see Section 6 – Workflow 1: Set Up & Explore the System Features.
Target platform	The portion of the SCS system used for productivity. Configurations created on the *development host* are loaded to the target platform for productivity.
User-provided	This refers to certain items that the user must provide at the user's expense. For more information, see Section 4 – User-Provided Items.

1.2 Product Literature

You can order product literature from the following Intel literature centers.

Table 2 – Product Literature

Location	Contact Information
U.S. and Canada	1-800-548-4275
U.S. (from overseas)	708-296-9333
Europe (U.K.)	44(0)1793-431155
Germany	44(0)1793-421333
France	44(0)1793-421777
Japan (fax only)	81(0)120-47-88-32

1.3 Reference Content

Contact your Intel Field Representative for assistance in acquiring the latest version of these documents.

Table 3 – Reference Documents

Document	Document No./Location
Intel® Industrial Solutions System Consolidation Series User Guide	This is the document you are reading now, available in PDF format on the *User Support* media USB that comes with the product packaging. (PN 538056)
User Support Documents	See the full listing of user support documents at Section 3 – Intel® Industrial Solutions System Consolidation Series Documentation.
Other product documentation	Get other product documentation by consulting with www.intel.com and/or your Intel Corporation sales representative.
Make certain to check the product support website for supplemental information and updates to this document.	www.intel.com/industrialconsolidation

§

2 Safety Notice

This safety notice summarizes information basic to the safe operation of the equipment described in this manual. The international symbol displayed below is a reminder that all safety instructions should be read and understood before installation, operation, maintenance, or repair of this product. When you see the symbol on other pages, pay special attention to the safety information presented. Observance of safety precautions will also help to avoid actions that could damage or adversely affect the performance of the product.

Do not attempt to perform any procedure before carefully reading all instructions. Always follow product labeling and manufacturer's recommendations. If in doubt as to how to proceed in any situation, contact your Intel Corporation representative.

2.1 Alerts for Warning, Caution, Important, and Note

2.1.1 WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. It may be used to indicate the possibility of erroneous data or device malfunction.

2.1.2 CAUTION

CAUTION indicates a potentially hazardous situation, which, if not avoided, may result in minor or moderate injury. It may also be used to alert against unsafe practices. It may be used to indicate the possibility of erroneous data or device malfunction.

2.1.3 IMPORTANT

IMPORTANT is used for comments that add value to the step or procedure being performed. Following the advice in the Important sections adds benefit to the performance of a piece of equipment or to a process.

2.1.4 NOTE

NOTE is used to call attention to notable information that should be followed during installation, use, or servicing of this equipment.

2.2 Safety During Installation and/or Maintenance

The target platform is designed to be repaired and serviced by a designated service representative. Any repair, servicing, or modification of this equipment that requires removal of any covers can expose parts and involves the risk of electric shock or personal injury. Make sure that the power switch is off and the product is disconnected from the main power source. Refer servicing to qualified personnel.

See further safety information in the documentation that comes with the target platform.

2.3 Safety Related to Data Preservation

CAUTION: Intel® Industrial Solutions System Consolidation Series provides development software that you will install onto a user-provided computer called a *development host*. Installation of the Intel® Industrial Solutions System Consolidation Series (SCS) software onto the development host will delete all existing hard drive contents. Make certain to back up any contents before installing the development host software. Failure to do so will result in loss of the data on the hard drive.

§

3 Intel® Industrial Solutions System Consolidation Series Documentation

For information about installing and operating Intel® Industrial Solutions System Consolidation Series (SCS), see:

- **Intel® Industrial Solutions System Consolidation Series (SCS) Startup Guide (PN 538055) –** This is a printed document that comes with the Intel® Industrial Solutions System Consolidation Series (SCS) package. This document also comes as an Adobe-formatted PDF document on the *User Support* USB media found within the Intel® Industrial Solutions System Consolidation Series (SCS) package.
- **Intel® Industrial Solutions System Consolidation Series (SCS) User Guide (PN 538056) –** This is the guide you are reading now. This is an Adobe-formatted PDF document that comes with the *User Support* USB media found within the Intel® Industrial Solutions System Consolidation Series (SCS) package.
- **MXE-5300 Series Fanless Embedded Computer User's Manual –** See this resource for information regarding the target platform hardware.
- **Wind River Development Host Help –** Access various help resources after booting the development host platform on the *Development Host* media USB. After booting and agreeing to the license information, the development host loads to a Linux OS environment. From there, go to **Applications > Wind River Documentation** and choose from the help resources available there.
- **Wind River Workbench Help –** Access various help resources after opening the development host software, Wind River Workbench. After clicking the Wind River Workbench icon, the development host system loads Workbench. Go to **Help** and choose from the information resources listed there. Further resources are available at www.windriver.com.

  NOTE: The development host includes a specially modified version of Wind River Workbench development software with features designed only for use with Intel® Industrial Solutions System Consolidation Series (SCS). The resources available from the Workbench help mentioned above may cover features not available on Workbench for SCS. For access to a full-featured version of Wind River Workbench, please contact Wind River Systems at www.windriver.com.
- **Intel Corporation Resources –** Make certain to contact with your sales and support representatives regularly. Also check for product updates, new products, and other resources on Intel's website at www.Intel.com.
- **Target Platform Documentation –** See *MXE-5300 Series 5301/5302/5303 Fanless Embedded Computer User's Manual* that comes as a printed document with the target platform *packaging*.

- **Licenses:** Access an important *Wind River Linux 5.0 Third Party License Notices* document by going to the following relative path on the development host media:

  …/home/wruser/WindRiver/legal-notices/wr-Linux-5.0.1/
  WindRiver_Linux5.0_ThirdPartyNotices_v2.1.pdf

  Access other license documents by going to the following relative path on the development host media:

  …/home/wruser/WindRiver/licenses

  IMPORTANT: Make certain to read important information regarding GNU General Public License version 3 (GPLv3) and GNU General Public License version 2 (GPLv2) in the *Wind River Linux User's Guide*, page 112. Access this document as an Adobe PDF file by going to the following relative path on the Development Host:

  …/home/wruser/WindRiver/docs/extensions/eclipse/plugins/com.windriver
  .ide.doc.wr_linux_5/wr_linux_users_guide/
  wr_linux_users_guide_5.0.1.pdf

§

4 User-Provided Items

4.1 Required User-Provided Items

The following sections define the required user-provided items for use with Intel® Industrial Solutions System Consolidation Series (SCS).

4.1.1 Development Host

While a fully configured target platform is able to operate as a standalone system, Intel Corporation recommends a direct and permanent connection between the target platform and the development host PC. A direct serial connection enables immediate monitoring of all guest operating systems on the target platform via the development host console.

Alternatively, after some configuration, users can log in to guest OSes via Ethernet, SSH, and telnet.

4.1.2 Development Host Specifications

CAUTION: SCS development software installs onto a *development host* computer that you provide. Installation of the SCS software onto a disk mounted on the development host will delete all existing storage disk (hard drive) contents. Make certain to back up any hard drive contents before installing the development host software. Failure to do so will result in loss of the data on the storage disk.

The Intel® Industrial Solutions System Consolidation Series (SCS) is comprised partly of a development host, which is hardware you must provide. The development host platform may be a desktop or laptop PC.

When selecting a development host PC, make certain that it meets or exceeds the specifications defined below:

- IBM PC-based PC, Core i3 processor
- 200 GB storage disk space
- 4 GB RAM
- One unused USB 2.0 port, USB 3.0 preferred
- Keyboard and mouse (for development host, if using a desktop PC)
- Keyboard (for target platform)
- Monitor capable of displaying 1024 x 768 @ 16 bpp or better (for development host)
- Monitor with DVI-I connector (for target platform)

- DB9 (RS-232) serial connector port
- One gender changer serial cable adapter
- One null modem serial cable adapter
- A network interface card (NIC), for debugging the target platform over Ethernet
- Internet connection

4.2 Optional User-Provided Items

The following are optional items you may provide while using SCS. This list is not all-inclusive.

- **Ethernet Connectivity –** Connect to the internet to receive critical updates and other information about SCS at www.Intel.com.
- **Uninterruptable power source (UPS) –** Depending on electrical supply reliability and quality, optionally connect the SCS chassis and development host to a power source protected by a UPS.

§

5 Development Workflows

The Intel® Industrial Solutions System Consolidation Series (SCS) development host uses *Wind River* Workbench** as its primary development tool. Some typical workbench activities or *workflows* are described in the following section.

5.1 Summary of Workflows

The Workbench platform can be used for innumerable development activities. This User Guide identifies several common development workflows to demonstrate the Workbench environment in action.

IMPORTANT: First read and then perform workflows in their chronological order within this User Guide.

The workflows are:

- Workflow 1: Set Up & Explore the System Features
- Workflow 2: Develop With Preloaded Workbench Projects
- Workflow 3: Build Linux & VxWorks OS Images from Scratch
- Workflow 4: Dynamically Load & Debug Applications on the Target Platform
- Workflow 5: Assign Persistent Target Platform OS IP Addresses
- Workflow 6: Configure a New Device

§

6 Workflow 1: Set Up & Explore the System Features

6.1 About This Chapter

Before you begin using Intel® Industrial Solutions System Consolidation Series (SCS) for development and production, use this chapter to familiarize yourself with the system.

IMPORTANT: This chapter is *Workflow 1*, the first of several *workflows* in this document. Make certain you *complete this workflow first* before proceeding with the other workflow chapters that follow. Perform the workflows in their order in this User Guide.

This chapter includes both system *background information* and *instructions*.

- FIRST… Read and understand this chapter entirely.
- THEN… Re-read the chapter and perform the instructions in the workflow.

6.2 System Introduction

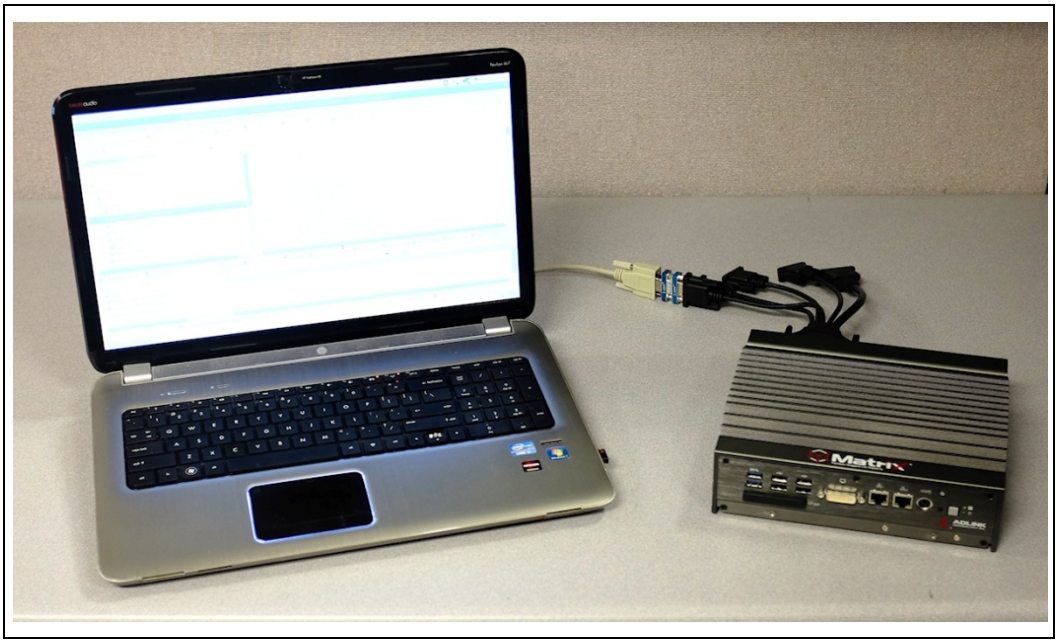
The Intel® Industrial Solutions System Consolidation Series (SCS) encompasses two subsystems: the *development host* (a PC provided by the end user) and the *target platform* (see Figure 1 – SCS Target Platform Box Shipment).

Figure 1 – SCS Target Platform Box Shipment

These two subsystems work together to provide you with development and productivity that you can customize for your needs.

As an example of the system in its simplest form, Figure 2 – Development Host and Target Platform System shows a laptop as a development host connected by a serial cable to the target platform. Optionally connect the subsystems remotely by Ethernet.

Figure 2 – Development Host and Target Platform System



NOTE: To render clarity in the figure, power supplies and external peripherals are removed. You may connect external peripherals such as a monitor and keyboard to enable interaction directly with the target platform via its Linux command-line interface.

Before you begin, make certain to read and understand this chapter prior assembling and exploring your Intel® Industrial Solutions System Consolidation Series (SCS) system.

6.3 User-Provided System Components

You must provide certain components to complete assembly of the SCS system:

- **Development host computer** perhaps as a desktop, laptop, or tablet PC. When selecting a development host PC, make certain that it meets or exceeds the specifications defined in Section 4.1.2 – Development Host Specifications.

NOTE: The SCS packaging provides a short serial cable dongle to connect the development host to the target platform. Depending on your development host's hardware, you may need also to supply a male-female serial connector to establish connection.

6.4 Target Platform Subsystem Overview

The SCS target hardware is a rugged software-controlled platform optimized for industrial productivity. See Figure 3 – The SCS Target Platform Hardware. For more information about the hardware, see the documentation within the target platform packaging.

Figure 3 – The SCS Target Platform Hardware

The SCS target platform comes preconfigured with a base set of software which you either can modify live via a remote connection with the development host, or by overwriting the system OSes with a bootable image (loaded to a USB) that you create using the development host.

SCS uses Wind River* Hypervisor 2 to control, monitor, and develop for the target platform's virtual machines. While the target platform hypervisor defines hardware access to specific virtual machines, the development host has a limited ability to modify access to virtual machine resources. For example, a network interface may be configured as visible only to the Linux platform while invisible to the other operating systems. You may learn to modify some of these hardware configurations, and this document provides instructions for this configuration in the workflow chapters that follow this workflow chapter.

The SCS target hard drive comes loaded with three virtual machine (VM) platforms as independent operating systems.

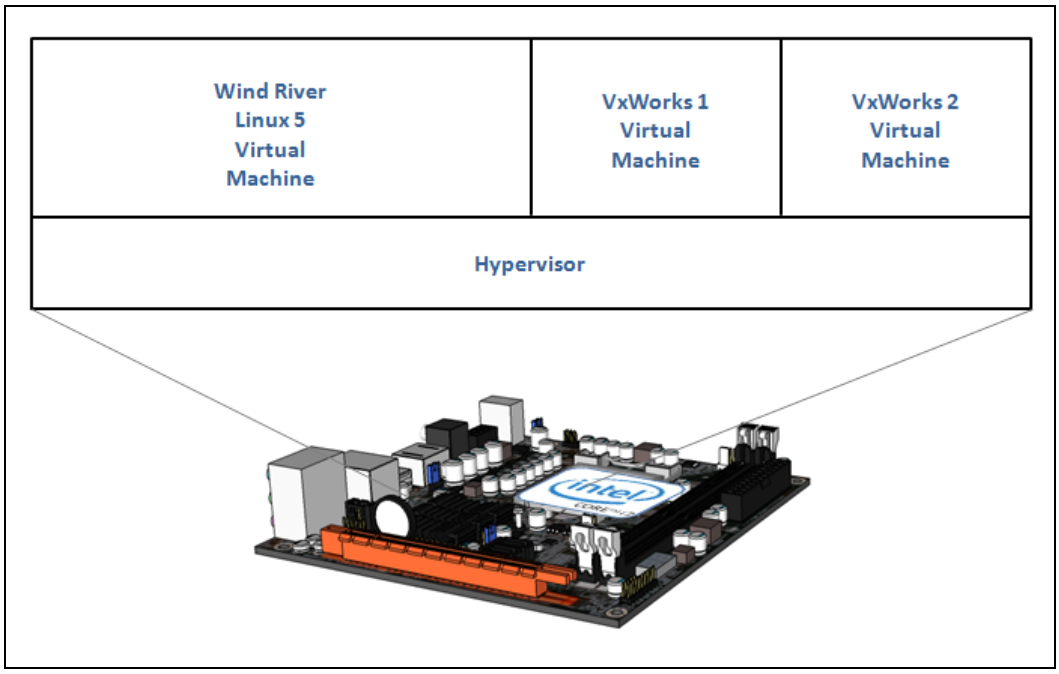
- One instance of Wind River* Linux* 5 OS
- Two instances of Wind River* VxWorks* 6.9 Real-Time OS

When you develop customized OS configurations to be run on the target platform, you may optionally boot the target platform from a USB that contains a bootable image carrying your customizations. Optionally copy this same image to the target platform's

hard drive via network or a serial connection to boot directly from the target platform hard drive.

Figure 4 – Target Platform OS Configuration and Figure 5 – Target Platform IO Configuration depict high-level views of the system.

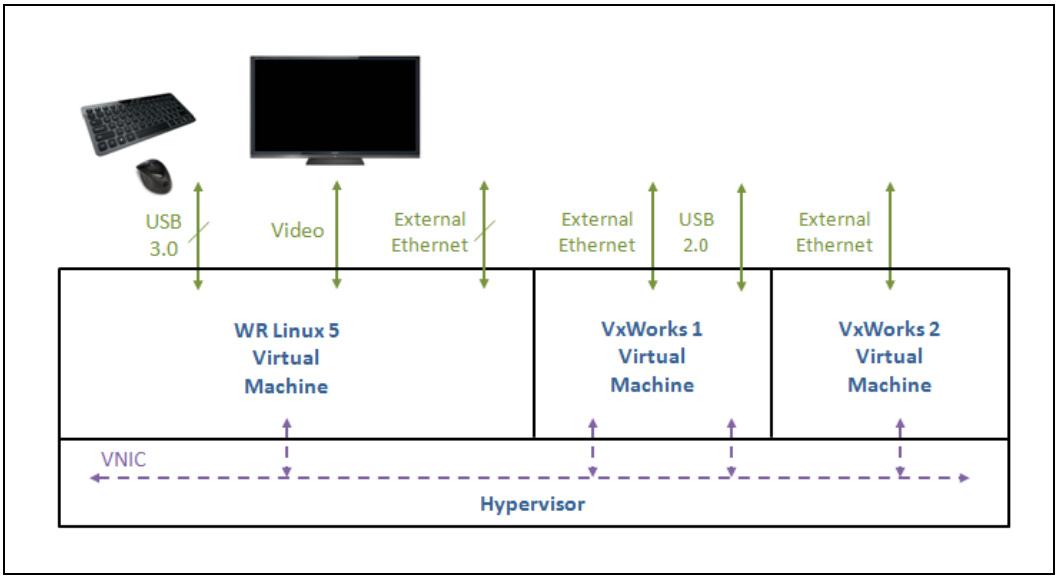
Figure 4 – Target Platform OS Configuration



6.4.1 Virtual NIC (VNIC)

The Intel® Industrial Solutions System Consolidation Series (SCS) four virtual OS platforms are networked together via a virtual layer 2 switch. Each individual virtual platform has a *virtual network interface card* (VNIC). This provides a virtual network infrastructure for the guest OSes to communicate with each other without using a physical network interface. See Figure 5 – Target Platform IO Configuration.

Figure 5 – Target Platform IO Configuration



NOTE: Dashed lines indicate virtual connections/devices.

The system cannot connect to the VNIC via an external interface directly. This means that when data is sent to the VNIC, it will always be transmitted to another Guest OS.

A VNIC appears to the guest OS as a standard Ethernet interface. However, unlike a real NIC, the data never leaves the target platform. The act of transmitting data from one VNIC to another is simply a memory copy of the Ethernet frame from one guest OS to another.

6.4.2 Default Target OS IP Address Assignment

Intel® Industrial Solutions System Consolidation Series (SCS) assigns target platform IP addresses to the three guest operating systems by default as follows:

- **Linux 5 –** 10.0.0.3
- **VxWorks 1 –** 10.0.0.4
- **VxWorks 2 –** 10.0.0.5

6.4.3 Customizable Target OS IP Address Assignment

While you can modify a guest OS address manually from a guest OS's command shell, this change is not persistent. A target platform reboot restores the default IP addresses (10.0.0.3 through 10.0.0.5). You can permanently override the guest OS boot IP addresses by configuring them using the development host Workbench software. To permanently change the IP addresses, see Section 10 – Workflow 5: Assign Persistent Target Platform OS IP Addresses.

6.4.4 Subsystem Connectivity

The single Linux VM and the two VxWorks VMs are configured to use the serial port to interface with the development host. Because there is a single physical serial port used for debugging, access is shared among the OSes. The asynchronous multiplexed I/O (AMIO) component takes care of multiplexing the serial streams. Wind River Workbench's AMIO terminals are designed to demultiplex the serial stream so that every platform's serial stream appears individually in its own Workbench console.

The system assigns the target platform devices (such as serial port, NIC, disk controllers, USB controllers, PCI cards) dynamically during the boot-up procedure.

6.4.5 Linux and VxWorks Operating Systems

The VxWorks VMs do not directly support human interface devices such as a monitor, keyboard and mouse. You must access these targets via a remote connection from the development host. The Linux VM does directly support keyboard, mouse and the on-board Intel graphics device. At this time it is only supported as a basic graphics terminal console. The Linux VM can also be accessed via a remote connection from the development host. You can do this either in the form of a target server connection or a SSH connection.

6.5 Development Host Subsystem Overview

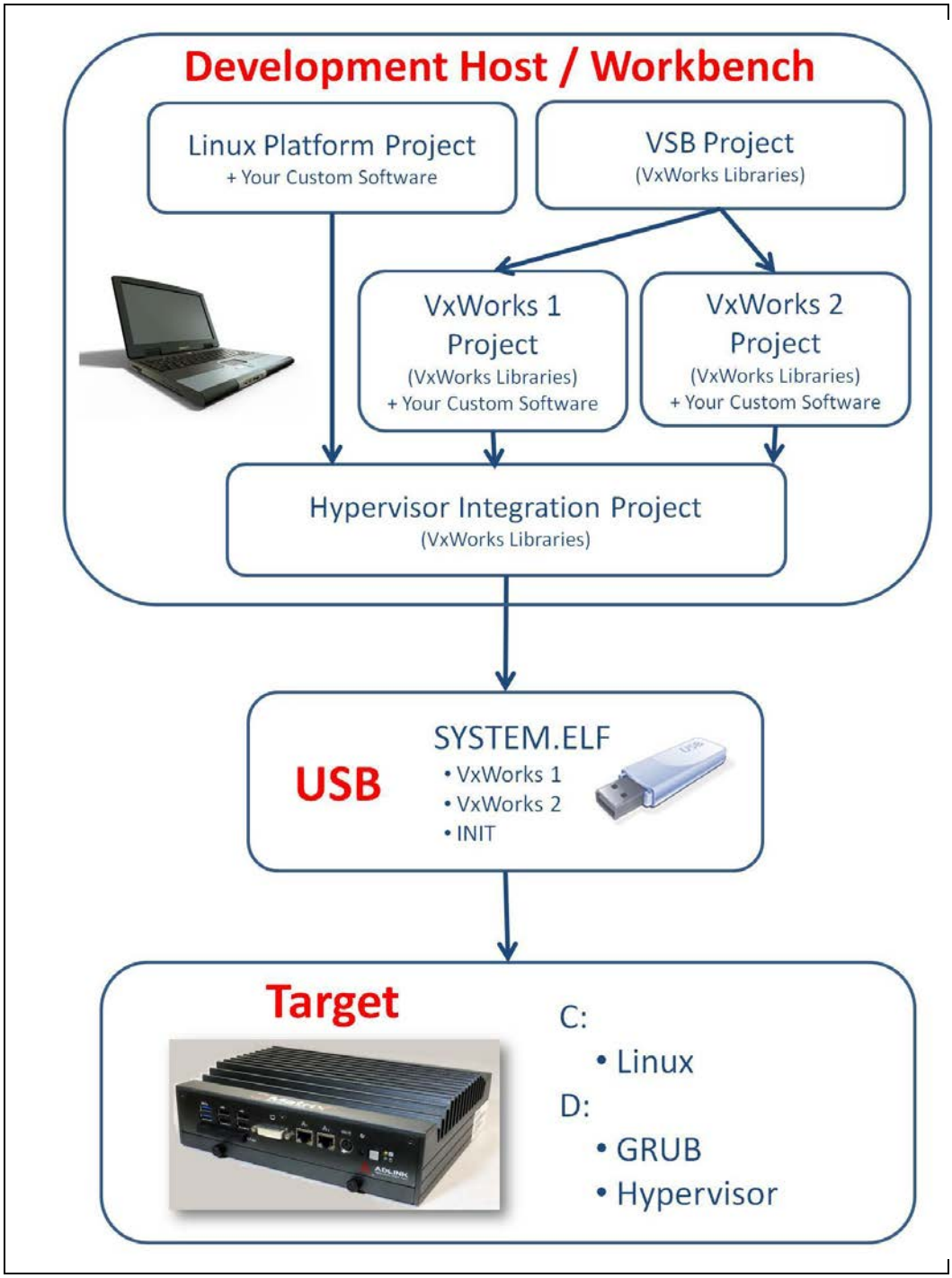
The SCS development host operates as a system separate from the target platform. The development host uses Wind River Systems Workbench software as its development tool. Within Workbench, you use special development units called *projects* to develop, organize, and compile customized productivity applications and other software.

After you finish development on the development host, you either:

- (Option A) load your developed software directly onto the target platform through a live connection (usually via serial cable or Ethernet), or
- (Option B) use development host *projects* to compile your software into Linux and VxWorks, which are then combined with other Workbench project information to create a bootable image file called **system.elf**. You then copy this image to USB media to boot the target.

  Figure 6 - Target Boot Image Project Compilation shows how the various development projects combine to create the bootable USB media image as described in Option B.

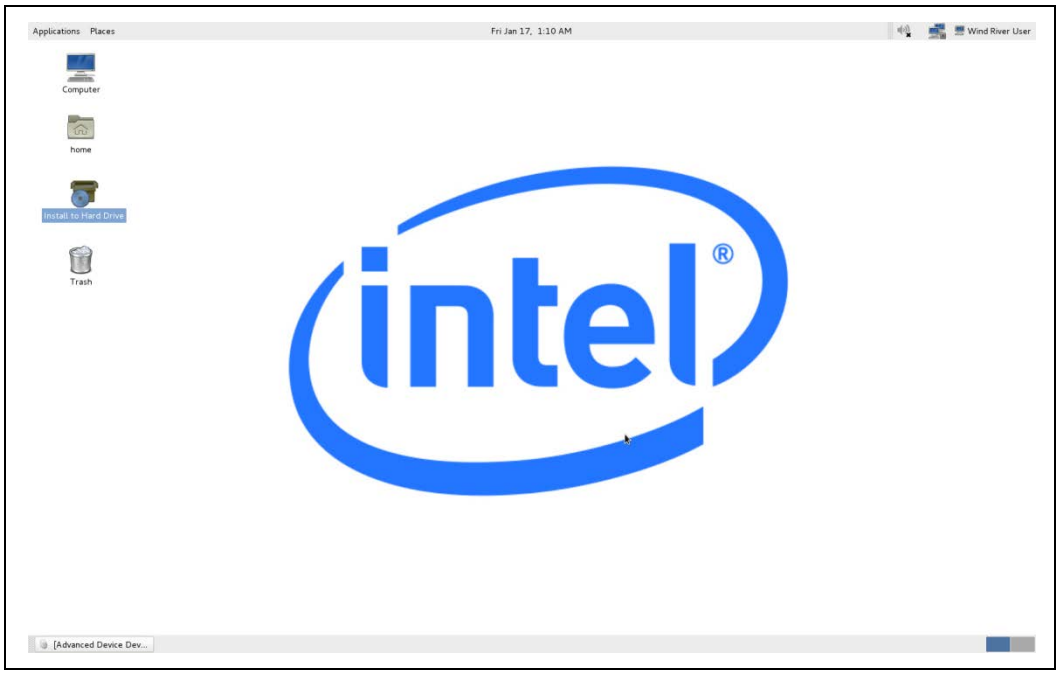
Figure 6 - Target Boot Image Project Compilation



6.5.1 Development Host Linux Shell

The development host Workbench software uses Linux as the resident platform for its development software. See Figure 7 - Development Host Desktop. When you start up the Workbench development host software, it loads on top of the development host Linux shell.

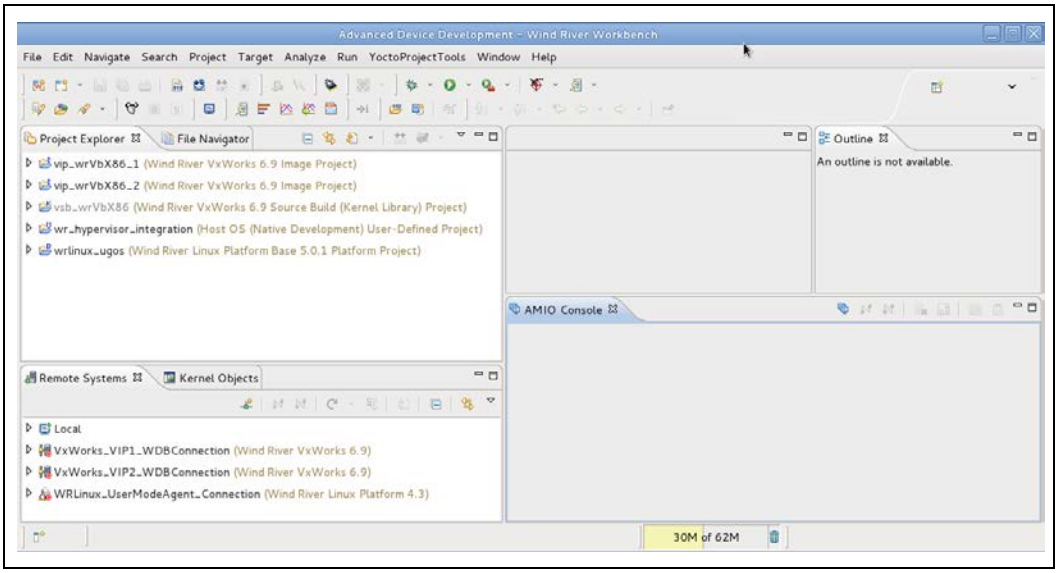
Figure 7 - Development Host Desktop



6.5.2 Wind River Workbench

The Intel® Industrial Solutions System Consolidation Series (SCS) development host uses *Wind River Workbench* as its primary development software. See Figure 8 - Workbench Workspace.

Figure 8 - Workbench Workspace


Advanced Device Development - Wind River Workbench
File Edit Navigate Search Project Target Analyze Run YoctoProjectTools Window Help
Project Explorer
File Navigator
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos (Wind River Linux Platform Base 5.0.1 Platform Project)
Outline
An outline is not available.
AMIO Console
Remote Systems
Kernel Objects
Local
VxWorks_VIP1_WDBConnection (Wind River VxWorks 6.9)
VxWorks_VIP2_WDBConnection (Wind River VxWorks 6.9)
WRLinux_UserModeAgent_Connection (Wind River Linux Platform 4.3)
30M of 62M

Workbench is an Eclipse-based development suite that provides an efficient way to develop real-time and embedded applications with minimal intrusion on the target system. It is an integrated development environment for creating software that runs on embedded Wind River Linux or VxWorks systems. Workbench includes a full project management facility in addition to a suite of tools for source code development, debug and analysis. It provides the capability to manage multiple processes and threads on Linux and VxWorks systems.

For more information about Workbench please refer to the Wind River documentation. Find other Wind River documents in Workbench under **Help > Help Contents > Wind River Documentation**.

6.5.3 Special Version of Workbench

Intel® Industrial Solutions System Consolidation Series (SCS) comes with a special implementation of Wind River Workbench that runs on development host hardware that you provide. This version of Workbench is optimized for development of applications that you load onto and run on the target hardware.

After installing the SCS development software (see Section 6.6 – License the System) to your development host hard drive, you are ready to begin development using Wind River Workbench.

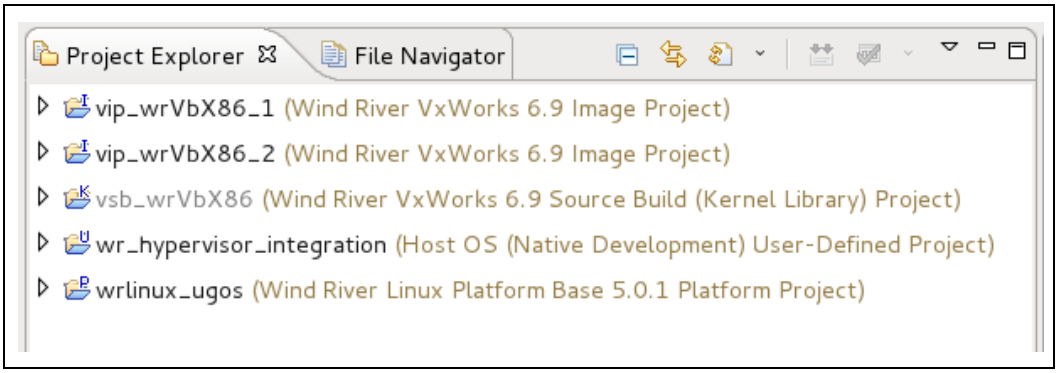
NOTE: For help using the special version of Workbench on SCS, read this *User Guide* in its entirety and access other help files on the User Support media.

6.5.4 Workbench Projects

The Workbench development environment uses *projects* as the building blocks for activities such as the development of applications for the target OSes.

Figure 9 - Project Explorer shows several pre-configured projects as they appear in Workbench's top workspace console, the *Project Explorer*.

Figure 9 - Project Explorer


Project Explorer
File Navigator
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos (Wind River Linux Platform Base 5.0.1 Platform Project)

The SCS version of Workbench has *preconfigured projects*. These projects are prebuilt packages that help reduce time you spend in development effort.

The development host system comes pre-loaded with five preconfigured projects:

- **vip_wrVbX86_1** – This is the VxWorks image project (VIP) that provides the build information for the first VxWorks virtual machine that appears on the target platform. This project is configurable, for example, to include source code for applications you have created to run on VxWorks.
- **vip_wrVbX86_2** – This is the VxWorks image project (VIP) that provides the build information for the second VxWorks virtual machine that appears on the target platform. This project is configurable, for example, to include source code for applications you have created to run on VxWorks.
- **vsb_wrVbX86** – This is a VxWorks source build library that provides the building blocks for the VIPs mentioned above.

  NOTE: The **vsb_wrVbX86** project is not configurable, and thus requires no user modification or interaction.
- **wr_hypervisor_integration** – This is a hypervisor integration project that combines the Linux and the two VxWorks images and creates the single hypervisor target image. This image is used to boot the target. This type of project has limited configurability. If you create your own Linux or VxWorks project, you can edit the makefile to use your project instead of the default one. You can also update the script files used to allocate hardware elements to the various VMs.

  NOTE: The **wr_hypervisor_integration** project is not configurable, and thus requires no user modification or interaction.
- **wrlinux_ugos** – This is a Wind River Linux platform project that provides the build information for the Linux virtual machine that appears on the target platform.

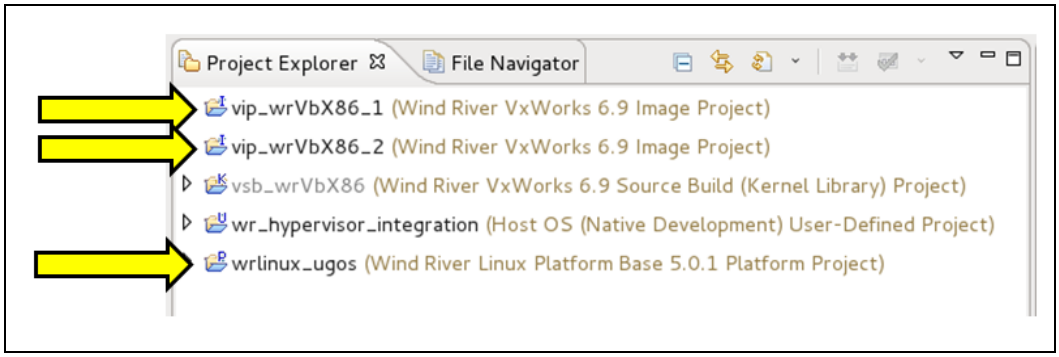
6.5.5 The Target Image

If you have your own application code, you can create a Linux application project, a VxWorks downloadable kernel module (DKM) project or a VxWorks real-time process (RTP) project to compile your code. Those projects can then be integrated in the Linux or VxWorks images.

NOTE: For more information about using DKMs in Workbench, see Section 8.4.3 – Add User Applications to the VIP Using DKM. Also see *Wind River Workbench User Guide Section 3.5*.

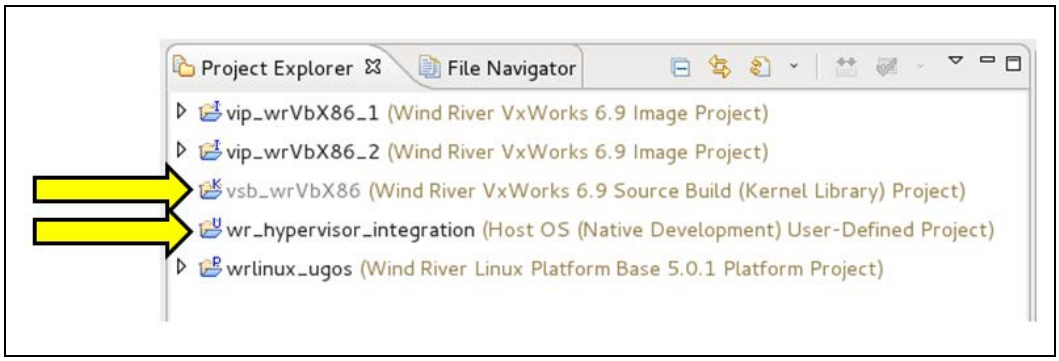
Using Workbench, you can also create your own Linux or VxWorks images instead of using the modifiable default projects shown by the arrows in Figure 10 – Modifiable Projects.

Figure 10 – Modifiable Projects


Project Explorer
File Navigator
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos (Wind River Linux Platform Base 5.0.1 Platform Project)

However, when creating a new target system, the SCS system will use two unmodifiable projects (**vsb_wrVbX86** and **wr_hypervisor_integration**) shown by the arrows in Figure 11 – Unmodifiable Projects.

Figure 11 – Unmodifiable Projects


Project Explorer
File Navigator
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos (Wind River Linux Platform Base 5.0.1 Platform Project)

Collectively, all Workbench projects created on the development host contribute to creating a bootable target image file specifically optimized to run on the SCS target.

After creating the target image file on the development host, you place this image file onto a bootable USB flash drive, insert the flash drive into the target platform, and then use the flash drive to boot the hypervisor and launch the guest OSes on the target platform.

Figure 6 - Target Boot Image Project Compilation shows that the VxWorks Source Build project combines with the two VxWorks Image Projects along with the Linux Platform Project. All of these projects funnel into the Hypervisor integration project (HIP). The HIP uses all project information to create the **system.elf** file, which you use to boot the VMs on the target hardware.

The hypervisor is preconfigured to boot three virtual machines: one instance of Linux and two instances of VxWorks. While you cannot modify this configuration, you can control which hardware devices (Ethernet Ports, Serial Ports, disk controllers, USB, and so on) are assigned to specific virtual machines that you configure using scripts you modify in Workbench. For more information about hardware resource assignment, see Workflow 6: Configure a New Device.

6.5.6 Development Host Hardware

The development host platform is hardware that you provide. For development host specifications, see Section 4.1.2 – Development Host Specifications.

6.5.7 Development Host Installation Software

Intel® Industrial Solutions System Consolidation Series SCS development host installation software resides on the *Development Host* media USB. Because the software has comes to you in an unlicensed state, follow the licensing procedure to activate the *Development Host* installation media. After activation, you may install the development host software onto your development host. See Section 6.6 – License the System.

6.6 License the System

Intel® Industrial Solutions System Consolidation Series is comprised of two subsystems:

- Target platform
- Development host

Read further for important licensing information pertaining to both subsystems.

6.6.1 Target Platform Software Licensing

The *target platform* is hardware shipped in the product packaging. While this requires no license acquisition on your part, please read, understand and comply with the licensing materials that come with the product packaging.

6.6.2 Development Host Software Licensing

The *development host* is hardware that you provide. Later you will install the software from the *Development Host* USB media onto the development host. Before you are able to install this software onto your development host PC, you must first request a license key file to activate your *Development Host* media.

IMPORTANT: Make certain your development host PC meets or exceeds the specifications shown in Section 4.1.2 – Development Host Specifications.

Follow this procedure to request a license key file.

6.6.2.1 Determine the Host-ID / Host Name

In this section, you determine your development host PC's -ID and Host Name.

CAUTION: After you determine your development host Host-ID and Host Name, you later install the development host software onto the development host PC you provide. When installing the development host software, the system destroys all data (including any applications and OSes) already installed on your development host PC hard drive. Make certain to back up all data to external storage media before installing the development host software. Failure to do so may result in loss of data on the development host.

Before you can license and activate your development host software, you must first provide some information specific to your development host PC. Part of the information you provide is your development host PC's:

- **Host Name –** A system name assigned to the development host PC.
- **Host-ID –** An 8 or 12 character hexadecimal number.

If your development host PC does not have a pre-existing OS…

If your development host PC does not have a pre-existing OS:

Insert the *Development Host* USB media into your development host and boot PC from that media.

After the system boots, a Linux development desktop appears. Follow the instructions in Section 6.6.2.1.3 - For Linux OS: to determine the development host Host-ID and Host Name.

If your development host PC does have a pre-existing OS…

Many times a development host is already running one of the operating systems mentioned in the following sections. Skim the sections that follow for instructions specific to your OS.

6.6.2.1.1 For Windows OS:

Use this instruction if your development host PC has a Windows OS.

If your development host machine has a single Ethernet port, use the 12-character MAC address.

Go to **Start > Run**. A **Run** window appears.

Type **cmd** and press **Enter**. A command console appears.

Type **ipconfig –all** and press **Enter**.

The result should be similar to the following:

```
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix . :
Description . . . . . . . . . . . : 3Com 3C920 Fast Ethernet
Physical Address. . . . . . . . . : 00-06-5A-21-95-44
```

In this example, the Host-ID is 00065A219544 (after removing the dashes from the physical address)

Record the Host-ID.

At the command prompt, type **hostname** and press **Enter**. A Host Name value appears.

Record the Host Name and proceed to Section 6.6.2.2 –– Locate the Serial Number.

6.6.2.1.2 For Windows OS (Alternate Procedure):

Use this instruction if your development host PC has a Windows OS and if the previous Windows instructions did not work.

If your development host machine has multiple Ethernet cards, is a laptop that is connected with a docking station with its own Ethernet card, or does not have a NIC card, use the hard drive serial number pre-pended with “DISK_SERIAL_NUM=”.

Go to **Start > Run**. A **Run** window appears.

Type **cmd** and press **Enter**. A command console appears.

Type **vol** and press **Enter**.

The result should be similar to the following:

```
Volume in drive C has no label.
Volume Serial Number is C0FC-0C4E
```

In this example, the Host-ID is **DISK_SERIAL_NUM=C0FC0C4E**. Make certain to pre-pend the value with **DISK_SERIAL_NUM=** with any dashes and colons removed.

Record the Host-ID.

At the command prompt, type **hostname** and press **Enter**. A Host Name value appears.

Record the Host Name and proceed to Section 6.6.2.2 –– Locate the Serial Number.

6.6.2.1.3 For Linux OS:

Use this instruction if your development host PC has a Linux OS.

Determine the 12-character Ethernet address in Linux.

Run the first line shown below and see the result similar to that shown in the second line.

```
% /sbin/ifconfig -a | grep -i hwaddr
eth0... Hwaddr 00:06:5B:82:F4:5A
```

In this example, the Host-ID is 00065B82F45A (after removing the colons from the value following **Hwaddr**).

Record the Host-ID and proceed to Section 6.6.2.2 –– Locate the Serial Number.

NOTES:

- Do not use **hostid**. The Linux **hostid** command produces a 6- or 8-character result that is not valid as a Host-ID.
- Make certain your Ethernet (eth) device lists as **eth0**. If your PC lists its first Ethernet device as **eth1** or **eth2** (etc.), the **Hwaddr** value will not work in conjunction with a development host license key, and your development host license will fail to function. If the first Ethernet (eth) device does not list as **eth0**, then rename it as **eth0**, and redo the **ifconfig** command shown above.

6.6.2.1.4 For Solaris OS:

Use this instruction if your development host PC has a Solaris OS.

At the command prompt, run:

```
% hostid
% 83299eed
```

The 8-character value shown above is the PC's Host-Id.

Record the Host-ID that appears on your display.

At the command prompt, type **hostname** and press **Enter**. A Host Name value appears.

Record the Host Name and proceed to Section 6.6.2.2 – Locate the Serial Number.

6.6.2.2 Locate the Serial Number

The 12-digit serial number is located on the top left corner of the *Startup Guide* shipped with this product. Find and record this number for use in the next step.

6.6.2.3 Go To License Website

Go to the following website:

```
registrationcenter.intel.com
```

Enter your email address and the serial number that was included on the *Startup Guide*. If you have not registered with your email address at the Intel Registration Center, you will be asked to create a new login ID and password.

After entering the email address and serial number (and registering as a new user if needed), a new web page will appear.

Enter the following information into the matching fields on the web page:

1. First Name:
2. Family Name:
3. Company Name:
4. Your Job Title:
5. Address:
6. Country:
7. Phone:
8. Email Address:
9. Your Company's Industry:
10. Host-ID:
11. Host Name:

Click the **Submit** button.

NOTE: Repeat the procedure for each additional license that you need to request for additional separate development host PCs.

IMPORTANT: The license issued will only activate the development host software for the PC hardware HOST ID you submitted. This license will not allow installation of the development host software onto a different PC.

Within approximately 24 business hours, you should receive a response email with a license file attachment from IOT SW Licensing (iot.sw.licensing@intel.com). If you do not receive a response for your license request, please:

- check your email spam folder for a response message, OR
- contact your Intel Corporation sales representative for assistance.

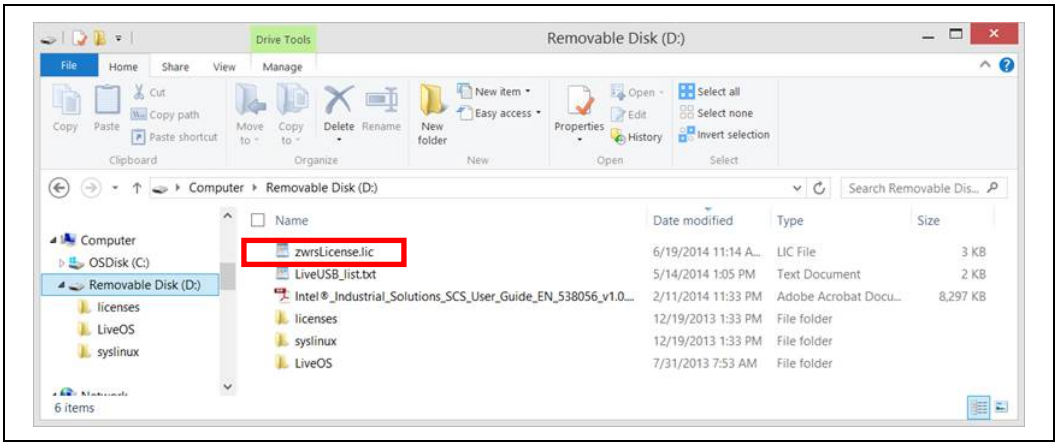
6.6.2.4 Install the License

After receiving the response email, follow the instructions below to install the license.

Insert the *Development Host USB* media into an available computer and copy the licensing attachment provided (wrhost.txt) in your licensing email to the root directory of the USB stick.

Rename the licensing file to zwrsLicense.lic. See Figure 12 – Rename the Licensing File. The example shown here is for a Microsoft* Windows* computer.

Figure 12 – Rename the Licensing File



Remove the USB stick.

Proceed to Section 6.7– Install the Development Host Software.

6.7 Install the Development Host Software

CAUTION: In this section you install the development host software onto the development host PC you provide. When installing the development host software, the system destroys all data (including any applications and OSes) already installed on your development host PC hard drive. Make certain to back up all important data to external storage media before installing the development host software. Failure to do so may result in loss of data on the development host.

After you license the development host flash media, follow this procedure to install the development software to your development host PC.

Power down the development host PC.

Insert the *Development Host USB* media into the user-provided development host PC.

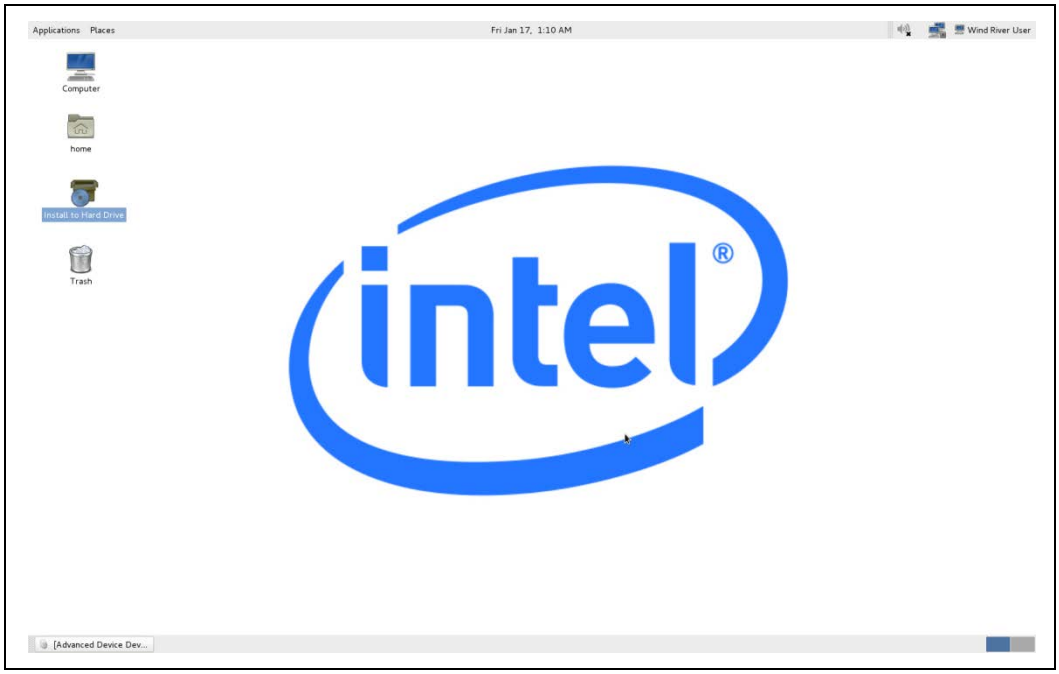
TIP: Intel Corporation recommends that you insert the media into a USB 3.0 port.

Power up the PC.

Several system screens appear and disappear as the system loads.

The license window disappears and the system loads to the Wind River Linux desktop. See Figure 13 – Linux Desktop.

Figure 13 – Linux Desktop

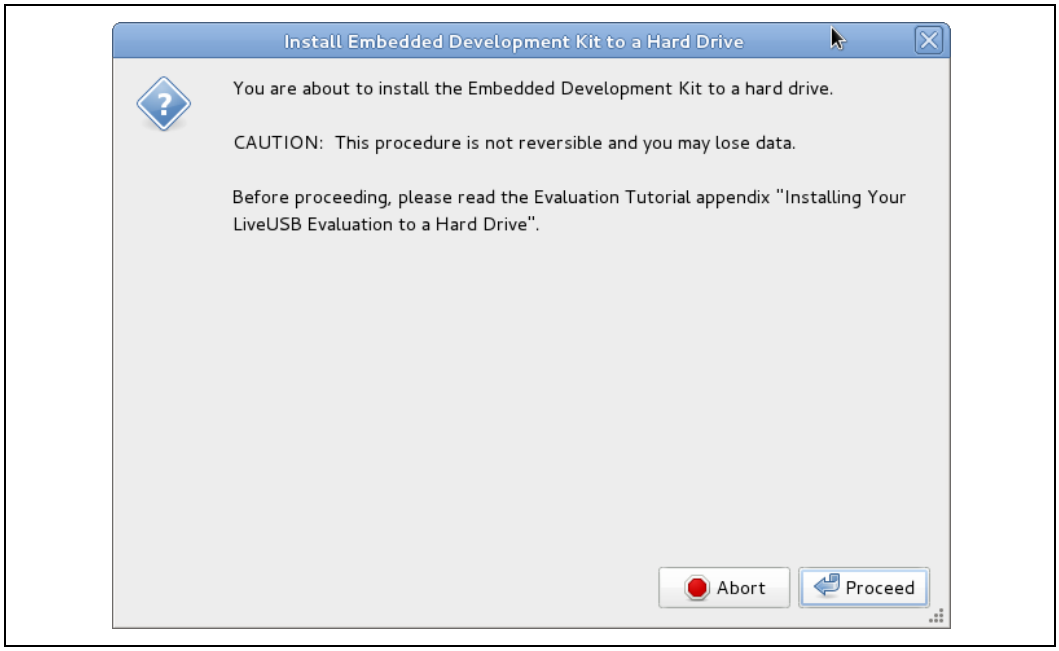


Double-click the **Install to Hard Drive** icon. See Figure 14 – Install Icon.

Figure 14 – Install Icon

A warning window appears. See Figure 15 – Warning Window.

Figure 15 – Warning Window


Install Embedded Development Kit to a Hard Drive
You are about to install the Embedded Development Kit to a hard drive.
CAUTION: This procedure is not reversible and you may lose data.
Before proceeding, please read the Evaluation Tutorial appendix "Installing Your LiveUSB Evaluation to a Hard Drive".
Abort
Proceed

Click **Proceed**. The cursor changes to a spinning disk for a few moments to indicate system progress. The cursor then returns to its default arrow appearance and remains this way for a few minutes.

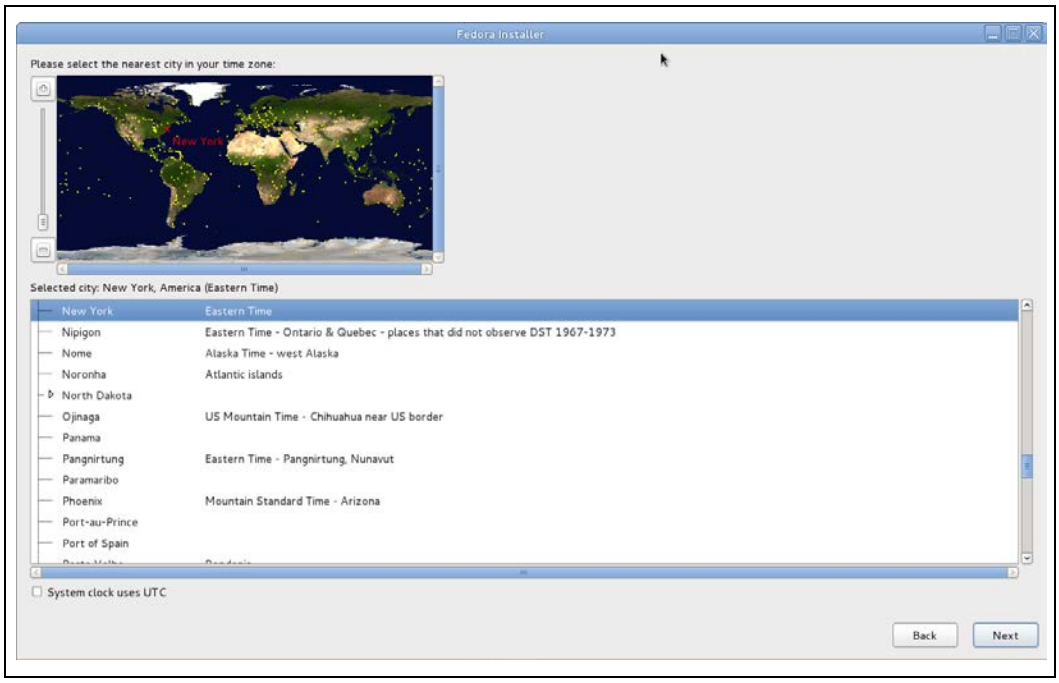
An **Examining Devices** window appears for a few moments. See Figure 16 – Examining Devices.

Figure 16 – Examining Devices

A Fedora installer window appears prompting you to select an interactive language. (Window not shown here)

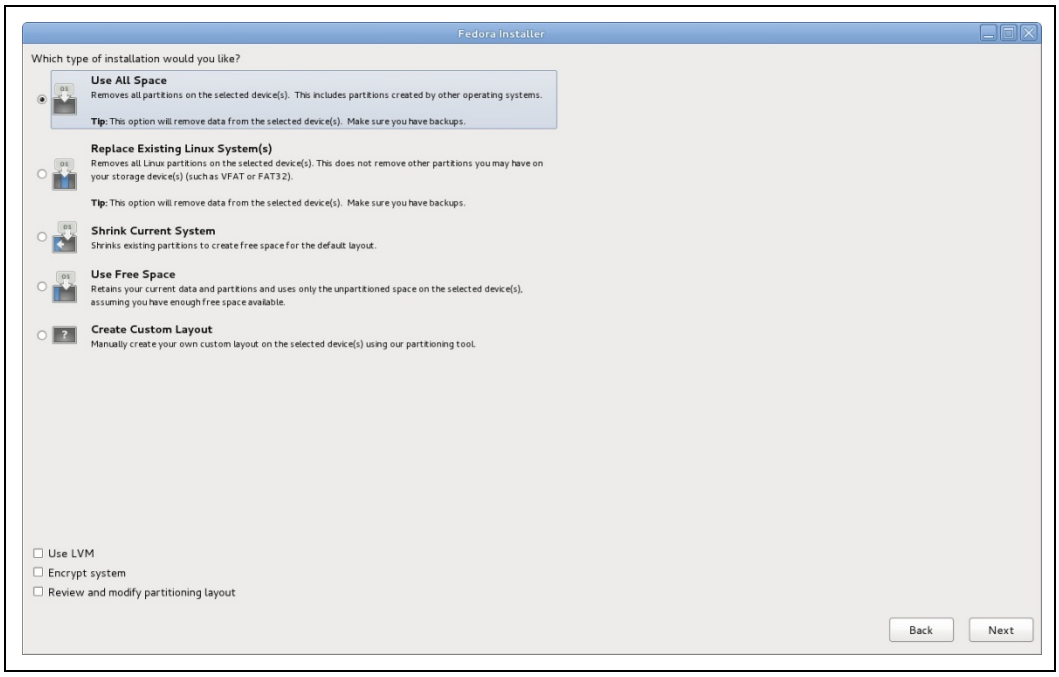
Select **US English** and click **Next**. The location window appears. See Figure 17 – Location Window.

Figure 17 – Location Window



Select your location and click **Next**. The following window appears. See Figure 18 – Installation Type.

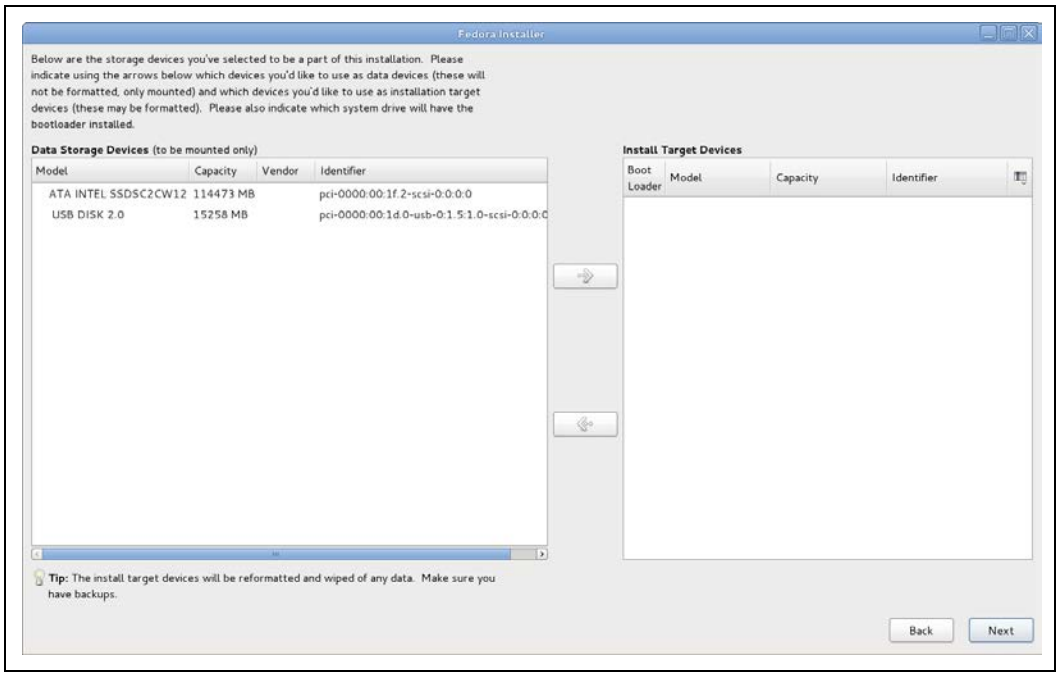
Figure 18 – Installation Type


Fedora Installer
Which type of installation would you like?
Use All Space
Removes all partitions on the selected device(s). This includes partitions created by other operating systems.
Tip: This option will remove data from the selected device(s). Make sure you have backups.
Replace Existing Linux System(s)
Removes all Linux partitions on the selected device(s). This does not remove other partitions you may have on your storage device(s) (such as VFAT or FAT32).
Tip: This option will remove data from the selected device(s). Make sure you have backups.
Shrink Current System
Shrinks existing partitions to create free space for the default layout.
Use Free Space
Retains your current data and partitions and uses only the unpartitioned space on the selected device(s), assuming you have enough free space available.
Create Custom Layout
Manually create your own custom layout on the selected device(s) using our partitioning tool.
Use LVM
Encrypt system
Review and modify partitioning layout
Back
Next

Select **Use All Space.**

Make certain not to check **Use LVM** and click **Next**. The storage space window appears. See Figure 19 – Storage Space.

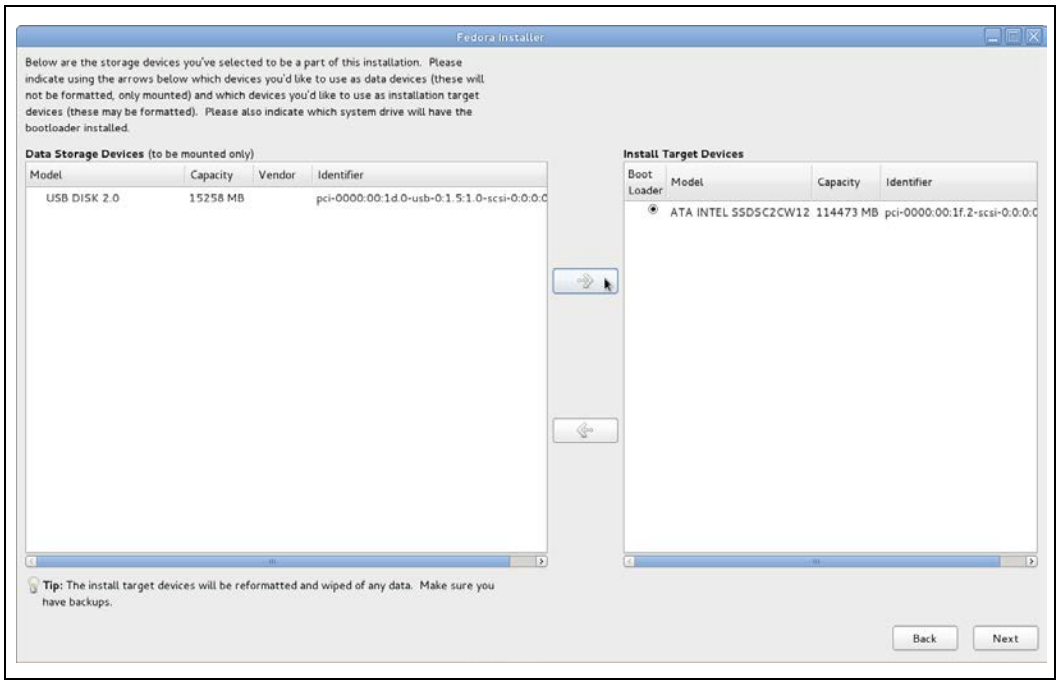
Figure 19 – Storage Space


Fedora Installer
Below are the storage devices you've selected to be a part of this installation. Please indicate using the arrows below which devices you'd like to use as data devices (these will not be formatted, only mounted) and which devices you'd like to use as installation target devices (these may be formatted). Please also indicate which system drive will have the bootloader installed.
Data Storage Devices (to be mounted only)
Model
Capacity
Vendor
Identifier
ATA INTEL SSDSC2CW12
114473 MB
pci-0000:00:1f.2-scsi-0:0:0:0
USB DISK 2.0
15258 MB
pci-0000:00:1d.0-usb-0:1.5:1.0-scsi-0:0:0:0
Install Target Devices
Boot Loader
Model
Capacity
Identifier
Tip: The install target devices will be reformatted and wiped of any data. Make sure you have backups.
Back
Next

In the **Data Storage Devices** pane (left) locate the local hard drive storage device onto which you wish to install the development host software. This adjacent figures use the **ATA Intel…** hard drive storage object as an example storage media.

Click the storage media object and then click the right (→) arrow to move the object into the **Install Target Devices** pane. See Figure 20 – Install Target Devices.

Figure 20 – Install Target Devices


Fedora Installer
Below are the storage devices you've selected to be a part of this installation. Please indicate using the arrows below which devices you'd like to use as data devices (these will not be formatted, only mounted) and which devices you'd like to use as installation target devices (these may be formatted). Please also indicate which system drive will have the bootloader installed.
Data Storage Devices (to be mounted only)
Model
Capacity
Vendor
Identifier
USB DISK 2.0
15258 MB
Install Target Devices
Boot Loader
Model
Capacity
Identifier
ATA INTEL SSDSC2CW12
114473 MB
Tip: The install target devices will be reformatted and wiped of any data. Make sure you have backups.
Back
Next

Click **Next**. A small **Examining Devices** window appears and the software begins to install. See Figure 21 – Examining Devices.

Figure 21 – Examining Devices

Numerous other process windows appear and disappear as the system installs. See Figure 22 – System Installs.

Figure 22 – System Installs

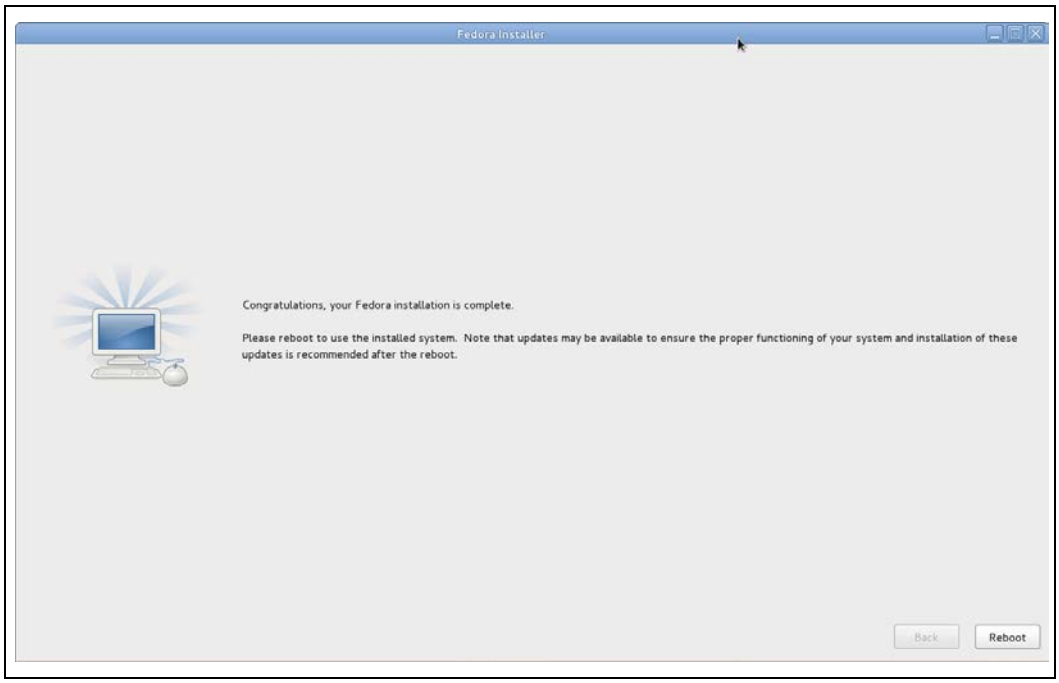
After a few minutes, the system prompts you, **Would you like to keep any changes you made to the LiveUSB environment?**

Select **No**.

NOTE: The system may take upwards of 45 minutes to install.

The system completes installing and a window appears prompting you to restart the system. See Figure 23 – Installation Complete.

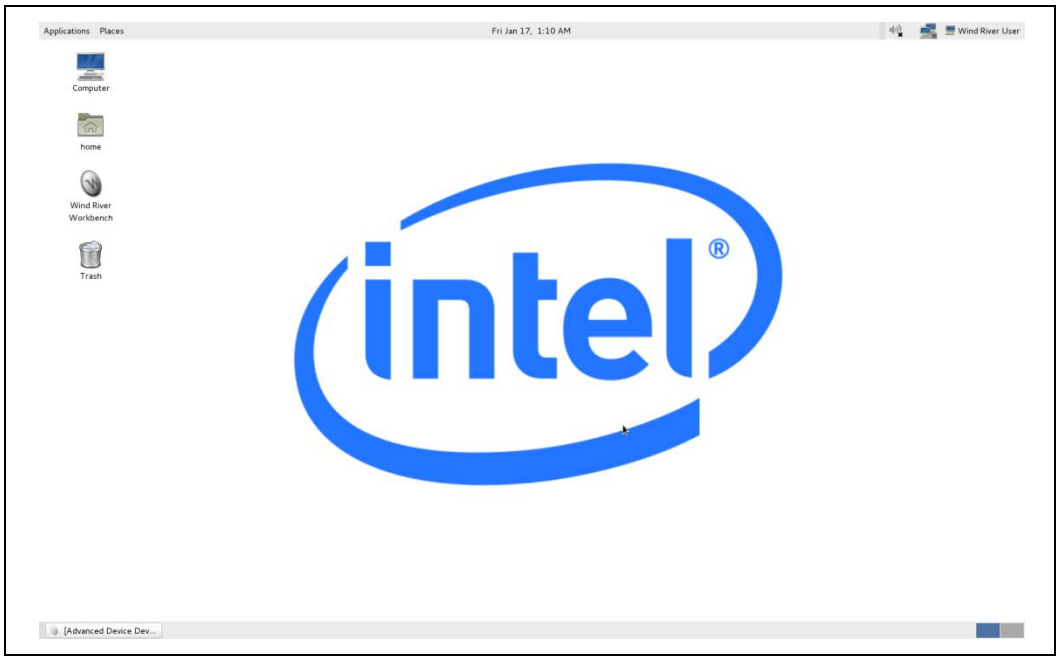
Figure 23 – Installation Complete


Fedora Installer
Congratulations, your Fedora installation is complete.
Please reboot to use the installed system. Note that updates may be available to ensure the proper functioning of your system and installation of these updates is recommended after the reboot.
Back
Reboot

Click **Reboot**. After the system shuts down, remove the *Development Host* USB media before the system restarts.

Allow the system to restart. The Wind River Linux development host desktop environment appears. See Figure 24 – Linux Desktop.

Figure 24 – Linux Desktop



6.8 Set Up the Target Platform & Development Hosts

Follow the procedure to set up the target platform and development hosts.

6.8.1 Set Up the Target Platform

Required tools:

- 1/8 inch (3 mm) flathead screwdriver

Use a 1/8 inch (3 mm) flathead screwdriver to attach the power inverter plug to the target power receptacle. See Figure 25 - Power Inverter Plug.

Figure 25 - Power Inverter Plug

Plug the power inverter power cord to a grounded power source receptacle.

IMPORTANT: Do not power up the target platform.

Connect the monitor to the target platform DVI-I video connector.

Connect the keyboard USB connector to a USB port on the target platform.

6.8.2 Connect the Target Platform and Development Hosts

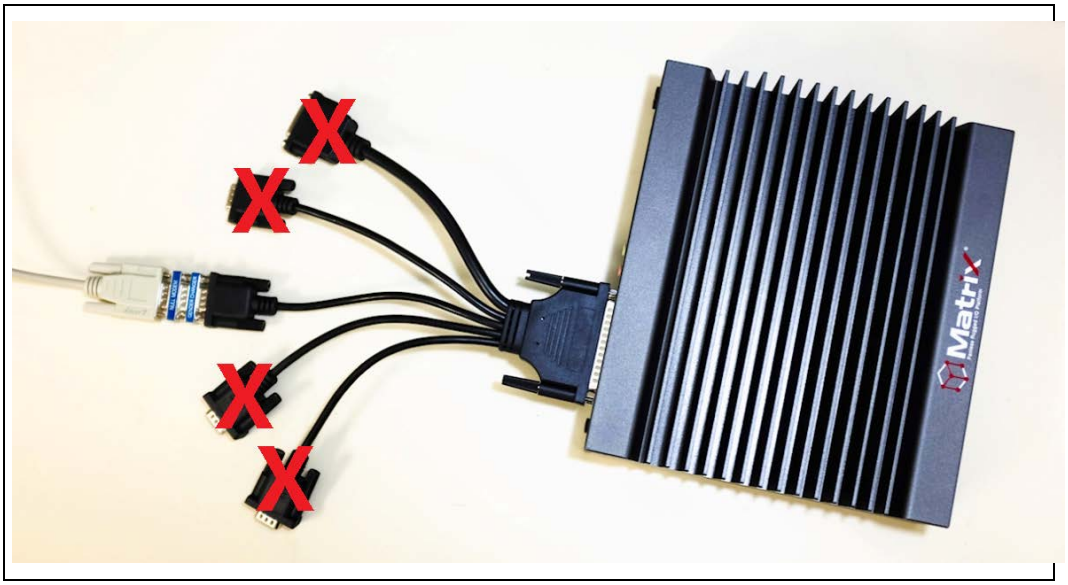
While there are several methods for the target platform and development host to communicate, the most common method is through a serial connection.

Follow the steps below to establish a serial connection.

Attach human interface devices to the development host. For a desktop PC, this includes the keyboard, mouse, and video/monitor cable. Connect the external video/monitor display to a grounded power source receptacle.

Connect the null modem serial cable between COM1 (labeled “1” on the target platform dongle) and the DB9 (RS-232) serial port on the development host. See Figure 26 - Target Platform Serial Connector and Figure 27 - Development Host Serial Connector.

Figure 26 - Target Platform Serial Connector

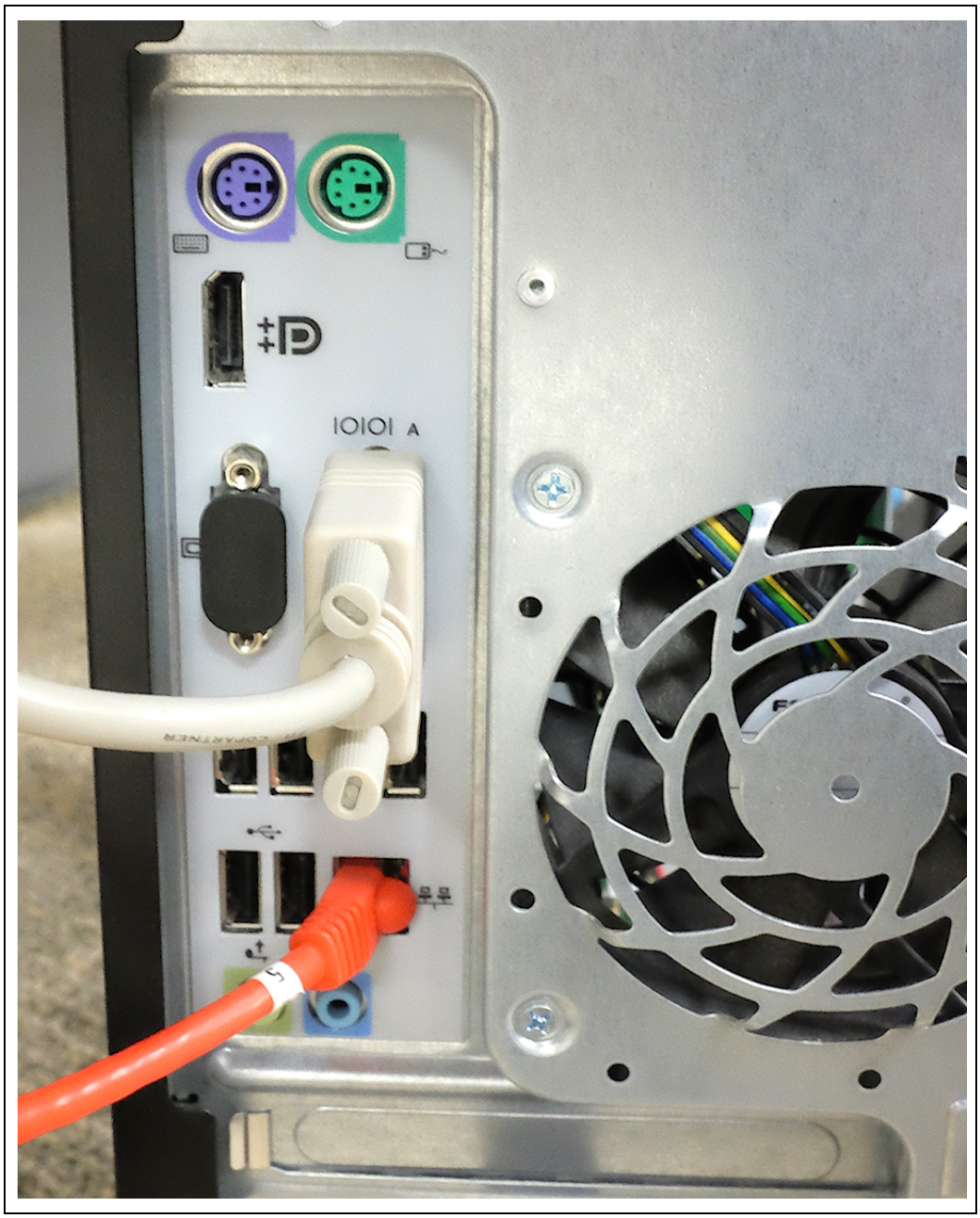


IMPORTANT: Large “X” marks in Figure 26 - Target Platform Serial Connector indicate serial ports you cannot use to connect the development host to the target platform. Make certain to use only the serial port connector labeled “1”.

NOTE: Optionally attach the target platform to an Ethernet network cable, and connect to the development host over Ethernet. Ethernet requires configuration to become functional, with either a static IP address or a DHCP configuration with a DHCP server on the network.

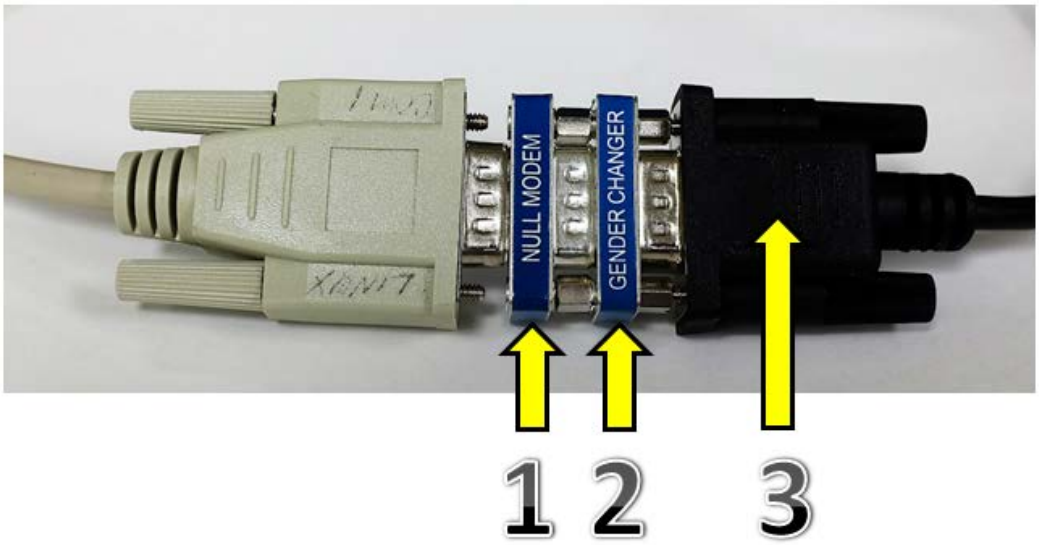
NOTE: The system requires that you provide development host hardware. Depending on the development host hardware you provide, the receptacle at the development host serial connection may vary from that shown in Figure 27 - Development Host Serial Connector. For information about the development host specifications, see Section 4.1.2 – Development Host Specifications.

Figure 27 - Development Host Serial Connector



NOTE: If you cannot establish a connection between the development host and target platform in the following steps, the serial cable connection between the development host and target platform may require the use of a null modem adapter and/or cable gender changer. See Figure 28 – Null Modem/Gender Changer.

Figure 28 – Null Modem / Gender Changer



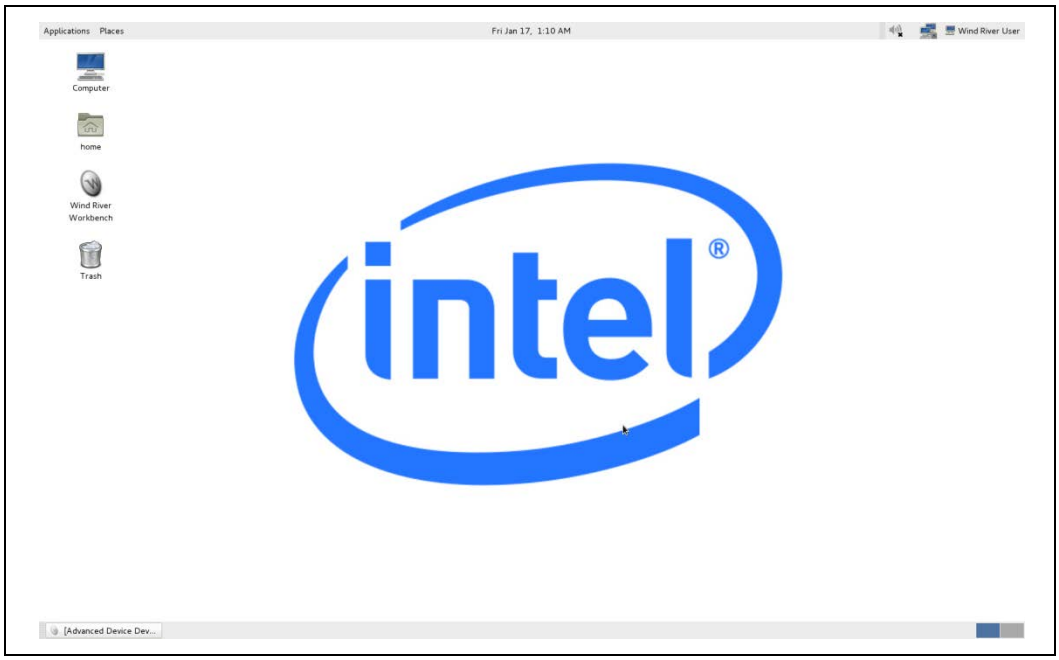
Item	Description
1	Null modem serial connector adapter
2	Gender changer serial connector adapter
3	Target platform DB9 (RS-232) serial port dongle labeled “1” at the location indicated by the arrow

Connect the development host power cord to a wall receptacle power source.

6.9 Explore the System

Start up the development host. The system loads to the Wind River Linux development host desktop. See Figure 29 - Development Host Desktop.

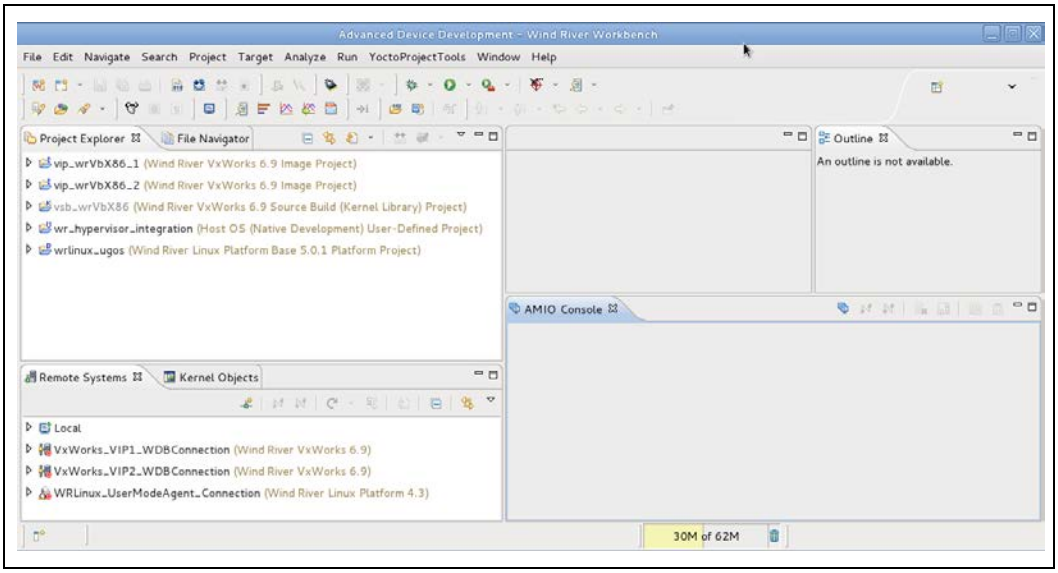
Figure 29 - Development Host Desktop



On the desktop, double-click the **Wind River Workbench** icon desktop icon.

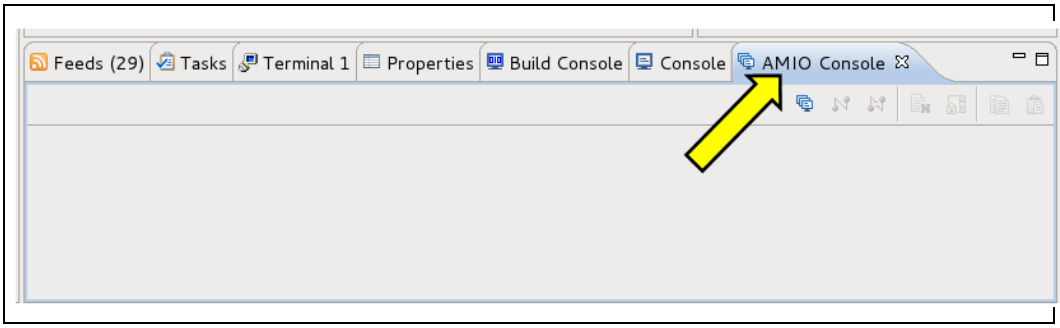
Wind River Workbench loads. See Figure 30 - Workbench Workspace.

Figure 30 - Workbench Workspace


Advanced Device Development - Wind River Workbench
File Edit Navigate Search Project Target Analyze Run YoctoProjectTools Window Help
Project Explorer
File Navigator
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos (Wind River Linux Platform Base 5.0.1 Platform Project)
Outline
An outline is not available.
AMIO Console
Remote Systems
Kernel Objects
Local
VxWorks_VIP1_WDBConnection (Wind River VxWorks 6.9)
VxWorks_VIP2_WDBConnection (Wind River VxWorks 6.9)
WRLinux_UserModeAgent_Connection (Wind River Linux Platform 4.3)
30M of 62M

From the top menu, select **Window > Show View > AMIO Console**. The AMIO console appears. See Figure 31 - AMIO Console.

Figure 31 - AMIO Console

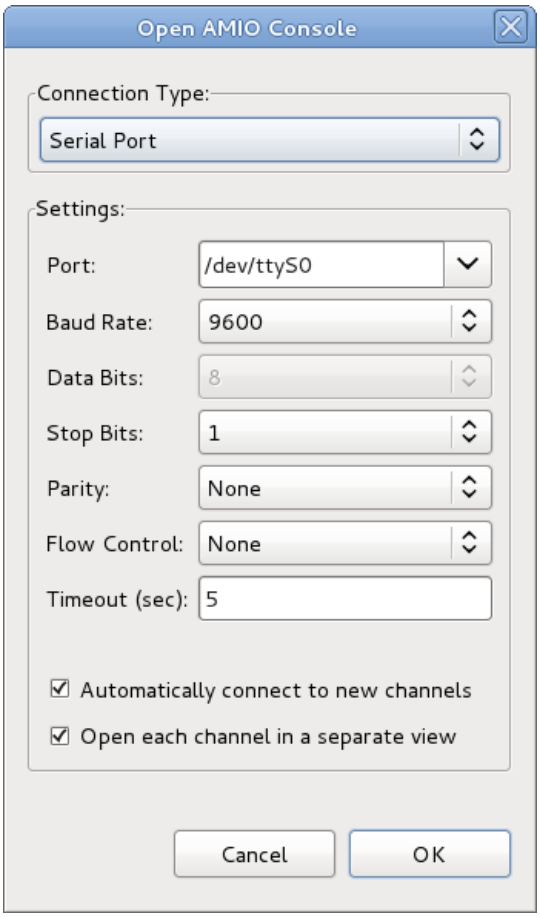


At the development host Workbench workspace, click the "Creates a connection for Application Multiplexed I/O" button. See Figure 32 - Multiplexed I/O Button.

Figure 32 - Multiplexed I/O Button

An **Open AMIO Console** window appears. See Figure 33 – Open AMIO Console.

Figure 33 – Open AMIO Console



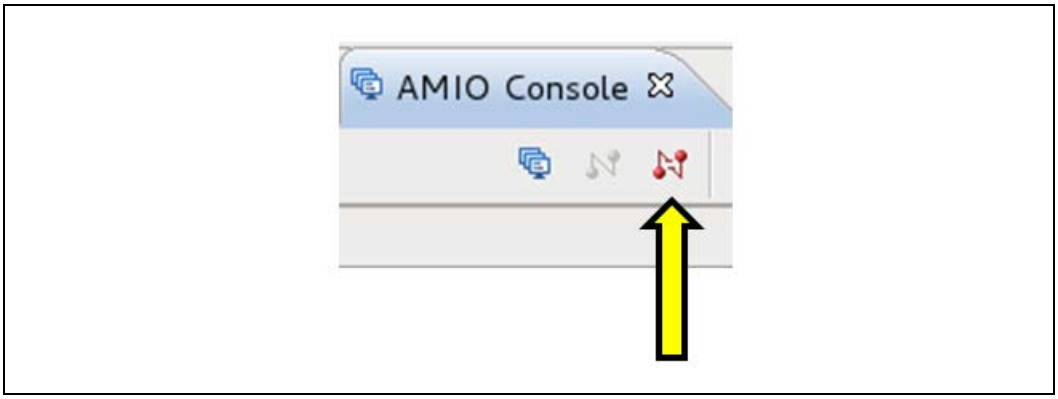
Select the parameters shown in Figure 33 – Open AMIO Console.

NOTE: If you are connecting through a USB-to-serial connector, you may not be able to connect using the **Port** parameter **/dev/ttyS0** above. If you are unable to connect using the **/dev/ttySO**, try connecting with **/dev/ttyUSB0**. Later references to this connection in this document assume you use the **/dev/ttyS0** connection.

Click **OK**. This opens up an AMIO connection at the development host serial port.

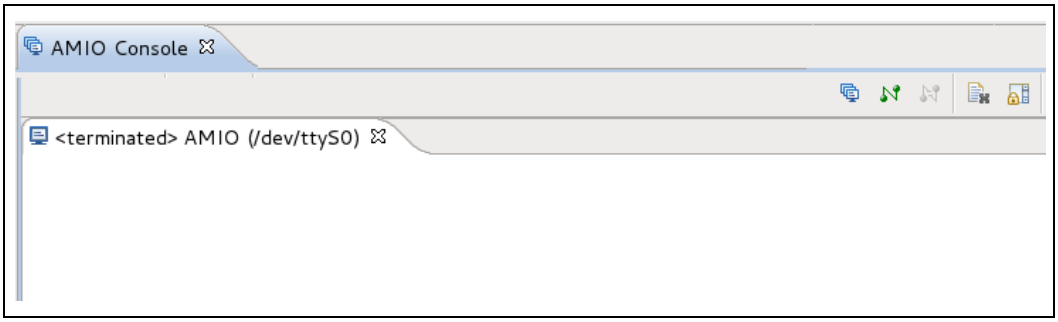
Near the bottom right section of the Workbench workspace, there is a grayed out "connected" icon adjacent to a red "disconnect" icon. See Figure 34 - Disconnect Icon. The red icon indicates the development host is listening for a connection to the target platform.

Figure 34 - Disconnect Icon



Optionally click the red disconnect icon to close the development host listening connection. After doing this, note that the green "connect" icon is enabled and the **/dev/ttyS0** console workspace shows the connection as **<terminated>**. See Figure 35 – Terminated Connection. Before continuing, make certain to re-click the green "connect" icon, and that the green icon has turned gray.

Figure 35 – Terminated Connection



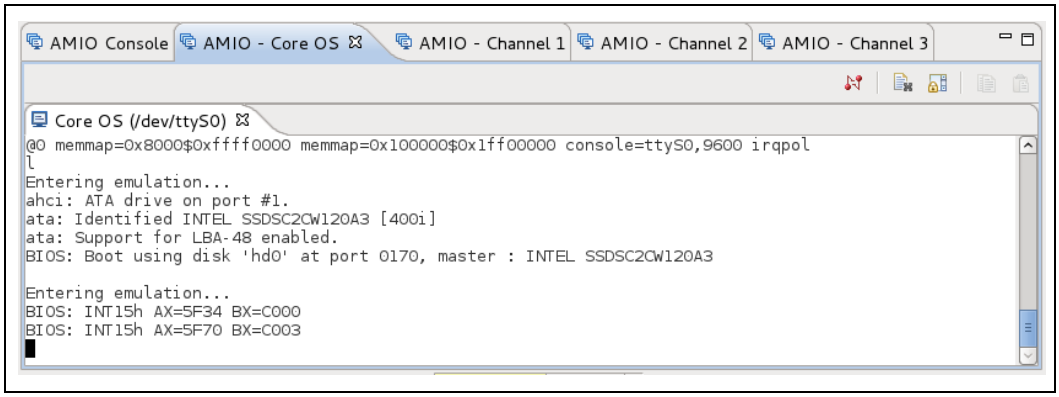
Double-click the **AMIO Console** tab. The AMIO console expands within the Workbench workspace.

Retrieve the *Target platform* USB media from the product packaging and insert it into a blue USB 3.0 port on the target platform.

Press and release the power switch on the target platform. The power indicator light on the target platform illuminates and the system starts.

As the system starts, screen output appears on the development host Workbench AMIO console. Separate AMIO consoles appear. Figure 36 - AMIO Consoles shows the console for the first instance of VxWorks. The figure also shows the tabs that correspond with each AMIO instance that appears.

Figure 36 - AMIO Consoles


AMIO Console
AMIO - Core OS
AMIO - Channel 1
AMIO - Channel 2
AMIO - Channel 3
Core OS (/dev/ttyS0)
@0 memmap=0x8000$0xffff0000 memmap=0x100000$0x1ff00000 console=ttyS0,9600 irqpol
l
Entering emulation...
ahci: ATA drive on port #1.
ata: Identified INTEL SSDSC2CW120A3 [400i]
ata: Support for LBA-48 enabled.
BIOS: Boot using disk 'hd0' at port 0170, master : INTEL SSDSC2CW120A3
Entering emulation...
BIOS: INT15h AX=5F34 BX=C000
BIOS: INT15h AX=5F70 BX=C003

NOTE: If not all AMIO consoles appear, then in Workbench go to **Window > Show View** and select the console you wish to view.

- **AMIO Console –** This console has no output. Optionally close this console.
- **AMIO - Core OS –** This displays hypervisor output.
- **AMIO - Channel 1** – This displays Linux OS output.
- **AMIO - Channel 2** – This displays output from the VxWorks OS 1.
- **AMIO - Channel 3** – This displays output from the VxWorks OS 2.

6.9.1 Default OS IP Address Assignment

Intel® Industrial Solutions System Consolidation Series virtual NIC (VNIC) assigns IP addresses to the guest operating systems by default as follows:

- **Linux –** 10.0.0.3
- **VxWorks 1 –** 10.0.0.4
- **VxWorks 2 –** 10.0.0.5

6.9.2 Inter-OS Communication over the Target VNIC

Using the physical (serial cable) connection between the development and the target platforms, you can use the Workbench AMIO console to display the logical connection among the Linux and VxWorks shells.

NOTE: To physically connect the development host to the target platform, see Section 6.8 - Set Up the Target Platform & Development Hosts.

The OSes on the target reside on a supervisory layer called a *hypervisor*, which also provides a platform for the target system's VNIC. The VNIC provides a network infrastructure that allows the OSes to communicate.

Follow this procedure to demonstrate communication among OSes over the target platform VNIC. Because you monitor this communication on the development host, this likewise demonstrates communication between the development and target platforms.

6.9.2.1 VxWorks-to-Linux Communication

To demonstrate that a communication path exists from VxWorks to Linux, visible from the VxWorks side, use the ping command.

Double-click any console tab. The console workspace expands within Workbench.

Click the VxWorks 1 (**AMIO – Channel 2**) console tab. The VxWorks (**AMIO – Channel 2**) console tab appears.

At the VxWorks command line, type **ping "10.0.0.3"** and press **Enter**.

A response indicates that Linux received the ping communication from VxWorks. See Figure 37 - VxWorks to Linux Communication.

Figure 37 - VxWorks to Linux Communication

```
-> ping "10.0.0.3"

Pinging 10.0.0.3 (10.0.0.3) with 64 bytes of data:
Reply from 10.0.0.3 bytes=64 ttl=64 seq=0 time=1ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 1 ms
rtt min/avg/max = 1/1/1 ms
value = 0 = 0x0
->
```

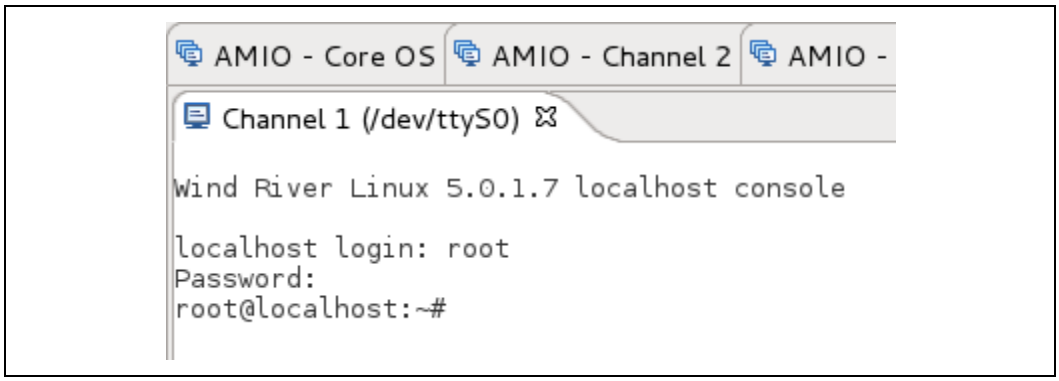
6.9.2.2 Linux-to-VxWorks Communication

To demonstrate that a communication path exists between Linux and VxWorks instances, use the ping command.

Go to the Linux (**AMIO – Channel 1**) console.

Login as **root** with the password **root**. The Linux command prompt appears. See Figure 38 – Linux Prompt.

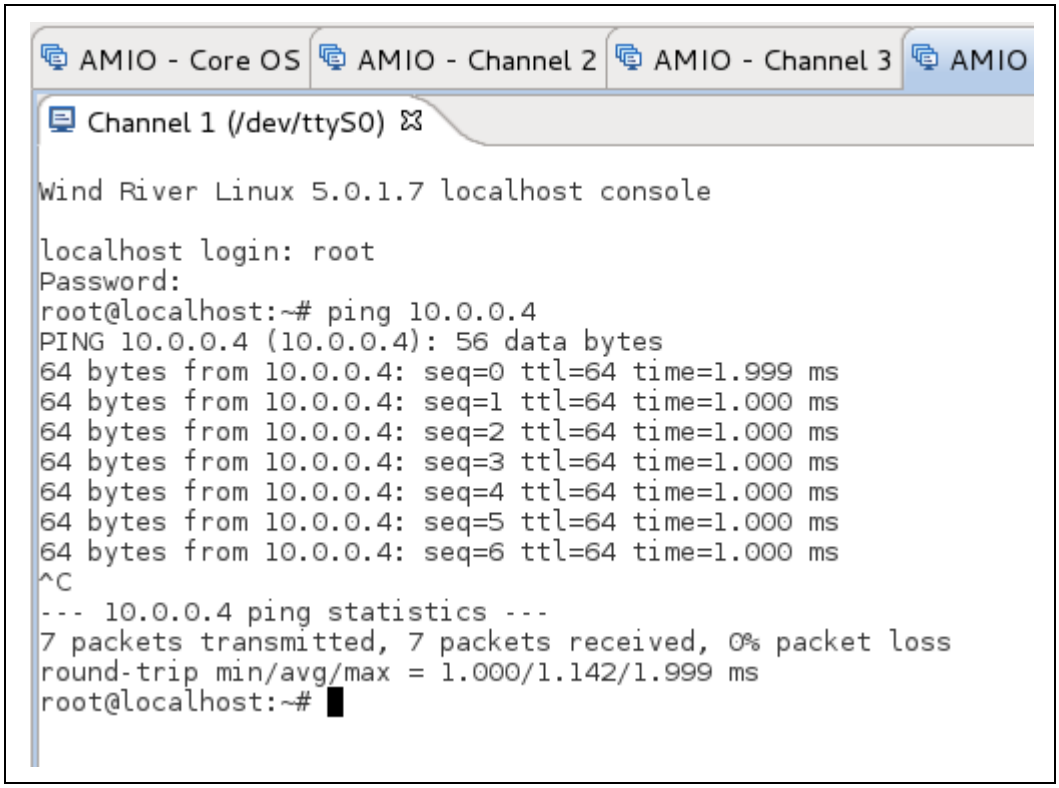
Figure 38 – Linux Prompt



At the command line, type **ping 10.0.0.4** and press **Enter.**

A response indicates that VxWorks 1 received communication from Linux. See Figure 39 - Linux-to-VxWorks Communication.

Figure 39 - Linux-to-VxWorks Communication


AMIO - Core OS
AMIO - Channel 2
AMIO - Channel 3
AMIO
Channel 1 (/dev/ttyS0)
Wind River Linux 5.0.1.7 localhost console
localhost login: root
Password:
root@localhost:~# ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: seq=0 ttl=64 time=1.999 ms
64 bytes from 10.0.0.4: seq=1 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=2 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=3 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=4 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=5 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=6 ttl=64 time=1.000 ms
^C
--- 10.0.0.4 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 1.000/1.142/1.999 ms
root@localhost:~#

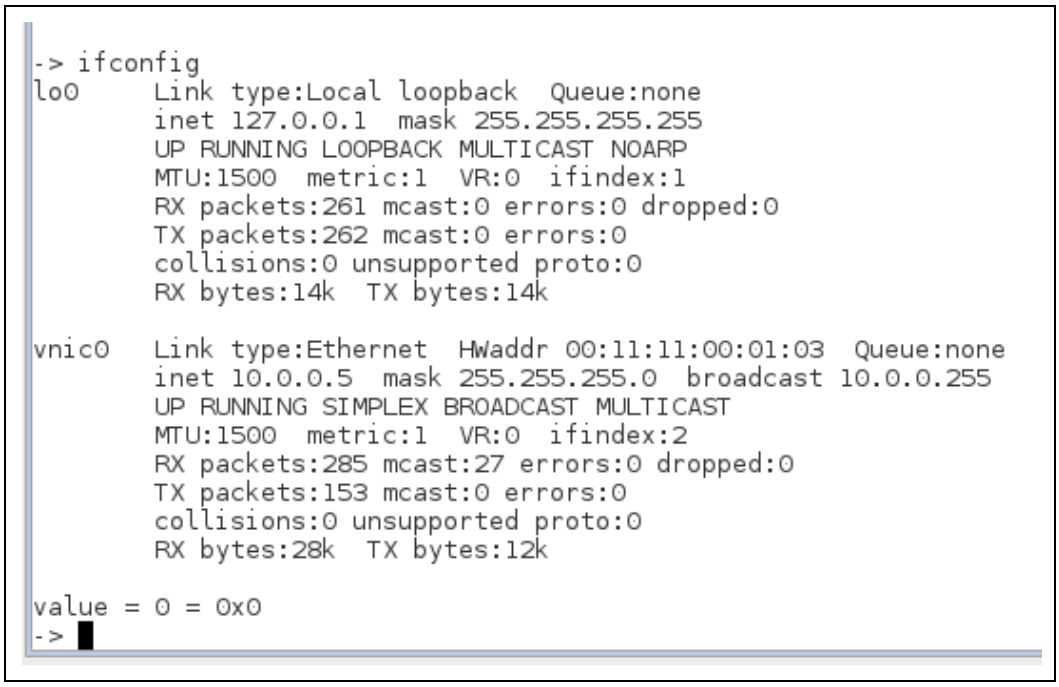
Press **CRTL+C** to stop the ping activity.

6.9.2.3 VxWorks-to-VxWorks Communication

Click the VxWorks (**AMIO - Channel 3**) console tab. The VxWorks (**AMIO - Channel 3**) console appears.

At the VxWorks prompt, type **ifconfig** and press **Enter**. Network information appears as output. See Figure 40 - Ifconfig Output.

Figure 40 - Ifconfig Output

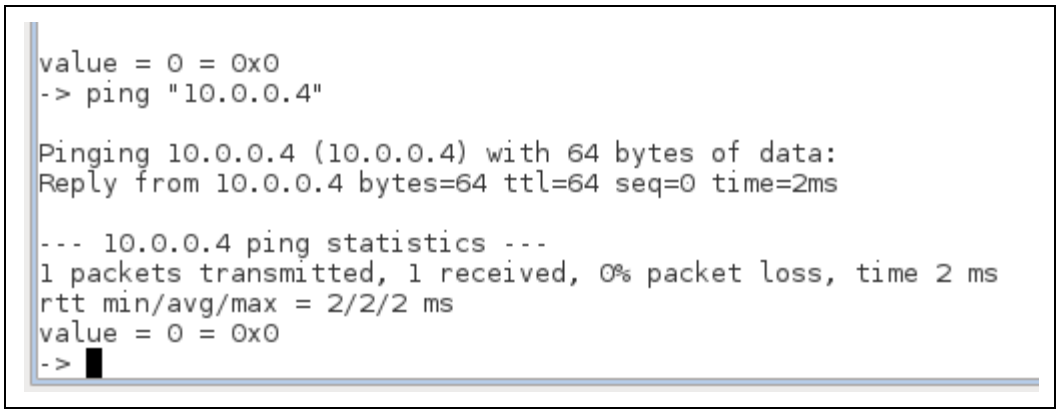

-> ifconfig
lo0 Link type:Local loopback Queue:none
inet 127.0.0.1 mask 255.255.255.255
UP RUNNING LOOPBACK MULTICAST NOARP
MTU:1500 metric:1 VR:0 ifindex:1
RX packets:261 mcast:0 errors:0 dropped:0
TX packets:262 mcast:0 errors:0
collisions:0 unsupported proto:0
RX bytes:14k TX bytes:14k
vnic0 Link type:Ethernet HWaddr 00:11:11:00:01:03 Queue:none
inet 10.0.0.5 mask 255.255.255.0 broadcast 10.0.0.255
UP RUNNING SIMPLEX BROADCAST MULTICAST
MTU:1500 metric:1 VR:0 ifindex:2
RX packets:285 mcast:27 errors:0 dropped:0
TX packets:153 mcast:0 errors:0
collisions:0 unsupported proto:0
RX bytes:28k TX bytes:12k
value = 0 = 0x0
->

The **10.0.0.5** content within the output indicates that you are at the VxWorks 2 prompt.

At the prompt, type **ping "10.0.0.4"** and press **Enter**.

A response indicates that VxWorks 1 received communication from VxWorks 2. See Figure 41 - VxWorks-to-VxWorks Communication.

Figure 41 - VxWorks-to-VxWorks Communication


value = 0 = 0x0
-> ping "10.0.0.4"
Pinging 10.0.0.4 (10.0.0.4) with 64 bytes of data:
Reply from 10.0.0.4 bytes=64 ttl=64 seq=0 time=2ms
--- 10.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2 ms
rtt min/avg/max = 2/2/2 ms
value = 0 = 0x0
->

Optionally perform this procedure from the VxWorks 1 Workbench AMIO console, while pinging the other VxWorks OS.

This concludes the overview and demonstration of the system's basic features and capabilities. To learn how to use the system in greater detail, proceed with the workflows in the chapters that follow.

§

7 Workflow 2: Develop With Preloaded Workbench Projects

7.1 About This Chapter

The Intel® Industrial Solutions System Consolidation Series (SCS) development host uses *Wind River Workbench* as its primary development tool. Use the Workbench development platform for innumerable development activities called *workflows*.

IMPORTANT: Before you perform this workflow, you must perform Workflow 1. Intel Corporation recommends performing all other workflows previous to this workflow.

This chapter includes both system *background information* and *instructions* for connecting and exploring basic system functionality.

- FIRST… Read and understand this chapter entirely.
- THEN… Re-read the chapter and perform the instructions in the workflow.

In this workflow, you use preloaded Linux and VxWorks (template) projects in Wind River Workbench to quickly develop a target platform image and boot it on the target hardware. This workflow helps provide you with basic skills using Workbench in order to reduce source build and compilation time. Once built, you then load the complete target image onto the target platform and boot it up. In this workflow you will have the opportunity to learn how to port your own source code to the VxWorks projects, which boot up from independent VM partitions on the target platform.

This workflow is the shortened version of Workflow 3: Build Linux & VxWorks OS Images from Scratch, which provides instruction for more advanced code importing, and for building new Linux and VxWorks VMs from-scratch.

7.2 About Build / Rebuild

Within this procedure you are prompted to either **Build** or **Rebuild** a project.

Build compiles only files and changes that have been modified since the last full build, while **Rebuild** forces the recompilation of the entire project. The system cannot be automatically detect modifications to certain types of files (like scripts). In such cases, a **Build** would not detect the changed file, but a **Rebuild** would.

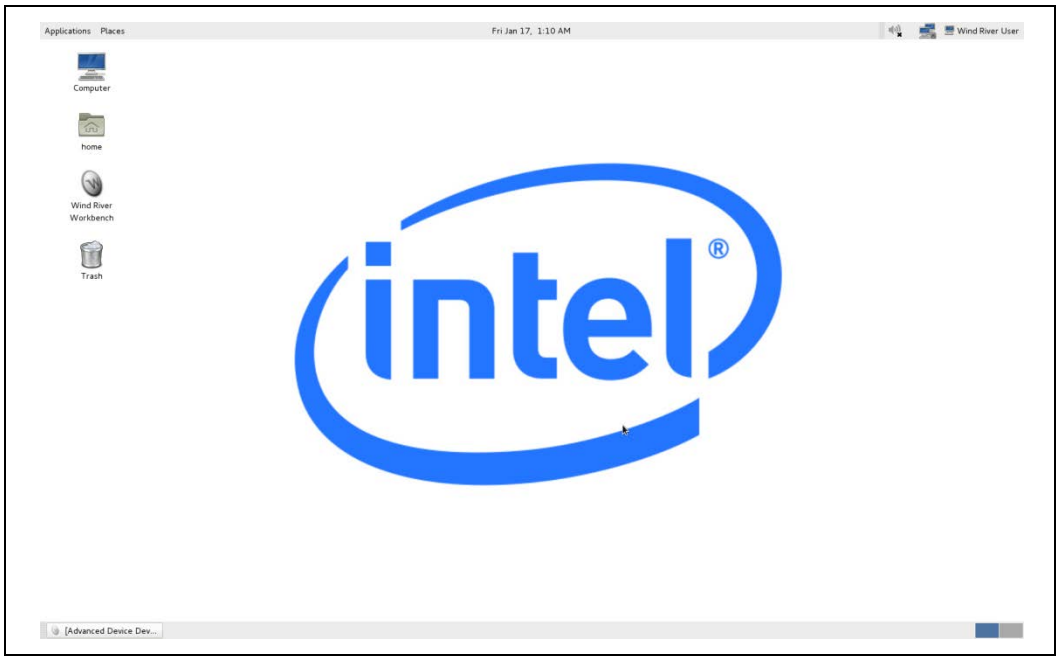
As a general rule, if you are modifying a source code file (such as .c or .h), a **Build** is appropriate. However, when changing another type of file, a **Rebuild** is the safer choice. Use **Rebuild** to ensure all changes are detected and recorded.

7.3 Start Up

Use the following steps after installing the development environment onto the development host's local hard drive.

Start the development host. The Wind River Linux desktop loads. See Figure 42- Linux Desktop.

Figure 42- Linux Desktop

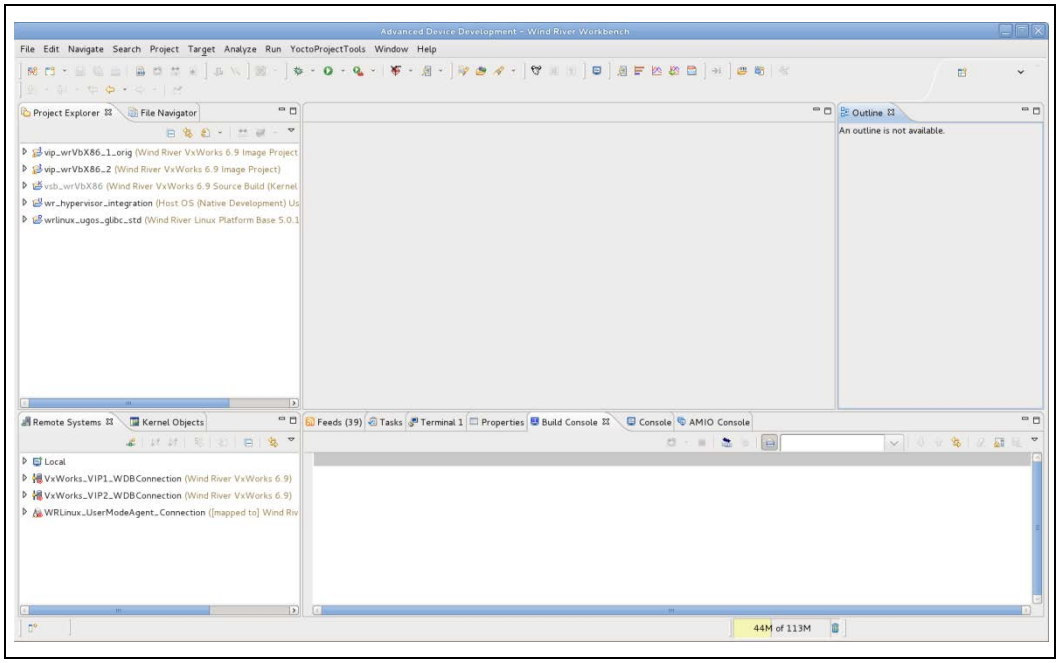


Open Workbench: Go to the desktop and double-click the **Wind River Workbench** icon. See Figure 43 - Workbench Icon.

Figure 43 - Workbench Icon

Wind River Workbench loads. See Figure 44 - Workbench Workspace.

Figure 44 - Workbench Workspace



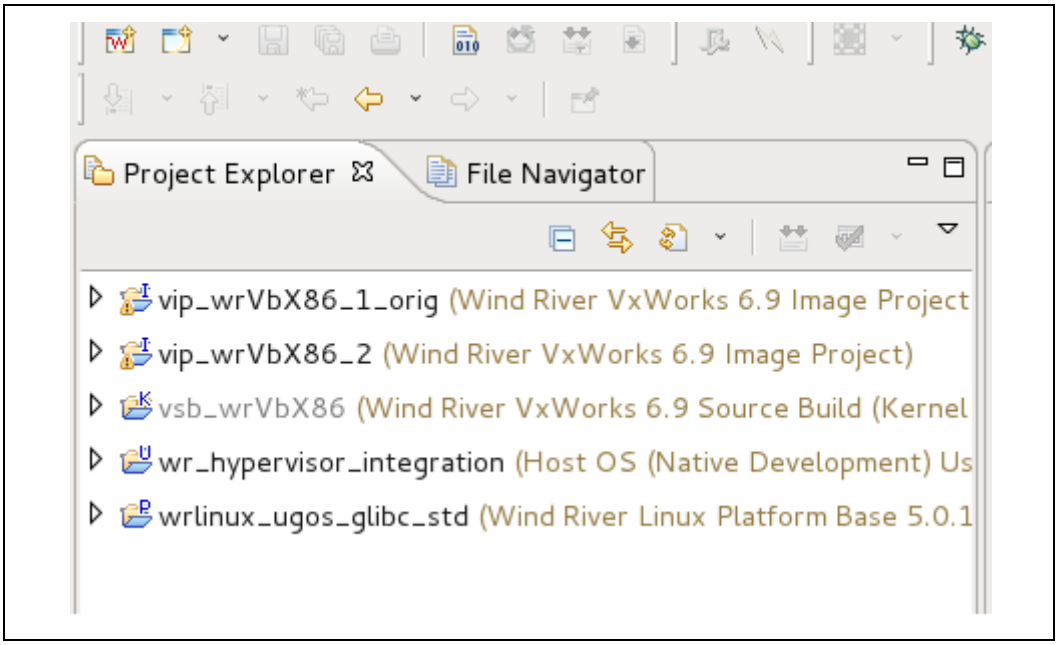
The *Project Explorer* is your main graphical interface for working with projects. You use it to create, open, close, modify, and build projects. You can also use it to add or import application code, to import or customize build specifications, and to access your version control system.

Using the Project Explorer, you can visually organize projects into structures that reflect their inner dependencies, and therefore the order in which they are to be compiled and linked.

7.4 Workbench Projects

The Workbench development environment uses *projects* as the building blocks for activities such as the loading of applications onto the target platform OSes. Figure 45 - Project Explorer shows several projects as they appear in Workbench's top workspace console, the Project Explorer.

Figure 45 - Project Explorer


Project Explorer
File Navigator
vip_wrVbX86_1_orig (Wind River VxWorks 6.9 Image Project
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel
wr_hypervisor_integration (Host OS (Native Development) Us
wrlinux_ugos_glibc_std (Wind River Linux Platform Base 5.0.1

Workbench uses *preconfigured projects*. These projects are prebuilt packages that help reduce time you spend in development effort.

The development host system comes pre-loaded with five preconfigured projects:

- **vip_wrVbX86_1** – This is the VxWorks image project (VIP) that provides the build information for the first VxWorks (VxWorks 1) OS that appears on the target platform. This project is configurable, for example, to include source code for applications you have created to run on VxWorks.
- **vip_wrVbX86_2** – This is a VIP just like the VxWorks 1 project (vip_wrVbX86_1) immediately above. It corresponds to the VxWorks (VxWorks 2) OS that appears on the target platform. This project is also configurable.
- **vsb_wrVbX86** – This is a VxWorks source build library that provides the building blocks for the VIPs mentioned above. This project is not configurable, and thus requires no user modification or interaction.
- **wr_hypervisor_integration** – This is a hypervisor integration project that combines the Linux and the two VxWorks images and creates the single hypervisor target image. This image is used to boot the target. This type of project has limited configurability. If you create your own Linux or VxWorks project, you can edit the makefile to use your project instead of the default one. You can also update the script files used to allocate hardware elements to the various VMs.
- **wrlinux_ugos_glibc_std** – This is a Wind River Linux platform project that provides the build information for the Linux virtual machine environment that appears on the target platform.

7.5 Modify Code in a Preconfigured OS Project

In this section you have the opportunity to write some simple source code into the preconfigured Workbench project image for VxWorks.

NOTE: You will have the opportunity to import code into the Workbench Linux project in a later workflow.

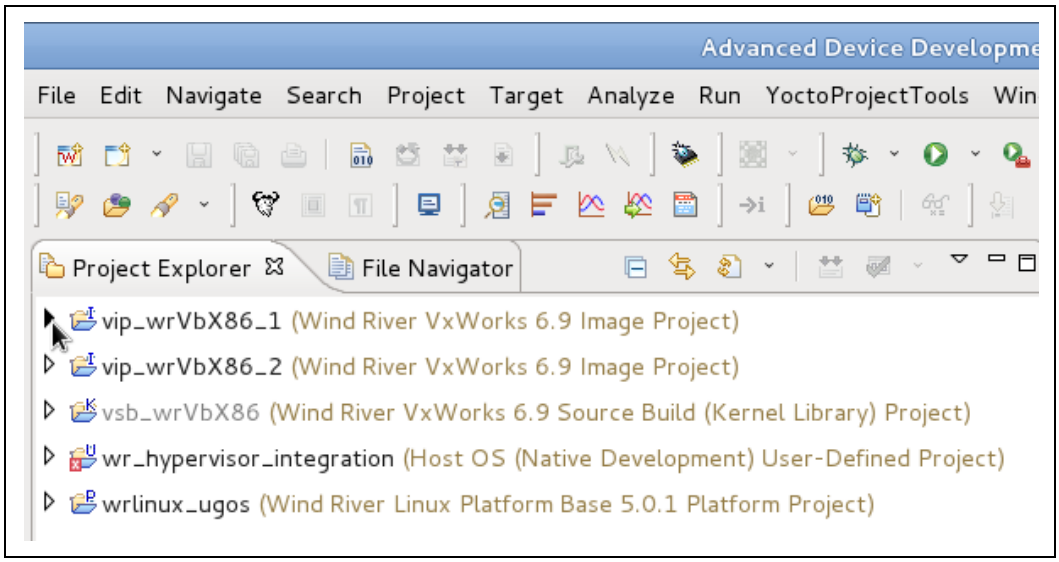
When you finish importing the code, you compile the projects, use Workbench to build the target platform boot image, copy it to USB, and boot the target platform from that image.

7.5.1 Modify Simple Code Line in VxWorks 1 Project

This procedure provides instruction for inserting a single line of code into the VxWorks 1 project.

In **Project Explorer**, expand the VxWorks project by clicking the triangular icon adjacent to **vip_wrVbX86_1**. See the darkened project icon in Figure 46 - VxWorks Project Icon.

Figure 46 - VxWorks Project Icon

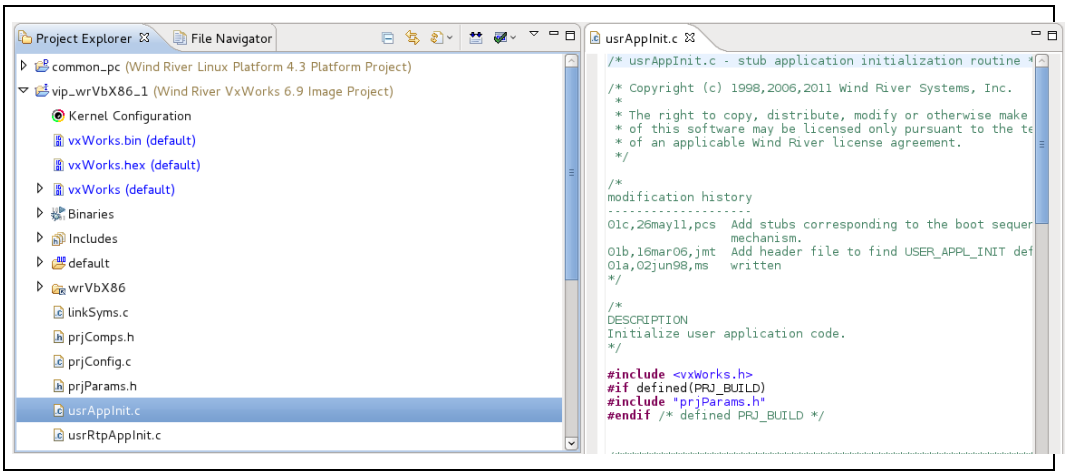

Advanced Device Developme
File Edit Navigate Search Project Target Analyze Run YoctoProjectTools Win
Project Explorer
File Navigator
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos (Wind River Linux Platform Base 5.0.1 Platform Project)

The VxWorks project expands.

In the listing that appears below the VxWorks line item, scroll down to the **usrAppInit.c** object.

Double-click the **usrAppInit.c** object. To the right of Project Explorer a **usrAppInit.c** work pane and **Outline** pane appear. See Figure 47 - usrAppInit.c.

Figure 47 - usrAppInit.c



Double-click the **usrAppInit.c** tab. The **usrAppInit.c** work pane expands.

NOTE: usrAppInit() is an application entry-point routine that can be modified to start your kernel application automatically at boot time. This is explained in further detail in the *VxWorks Kernel Programmer's Guide* which you can access through Workbench Help. Alternatively, for starting RTP applications refer to the *VxWorks Application Programmer's Guide* for details on using the usrRtpAppInit.c routine stub.

Within the **usrAppInit.c** work pane just below the banner at the top of the file locate the following line of code:

```
#include <vxWorks.h>
```

Insert the following just above the **<vxWorks.h>** include statement.

```
#include <stdio.h>
```

Place the cursor at the end of the code line below:

```
/* add application specific code here*/
```

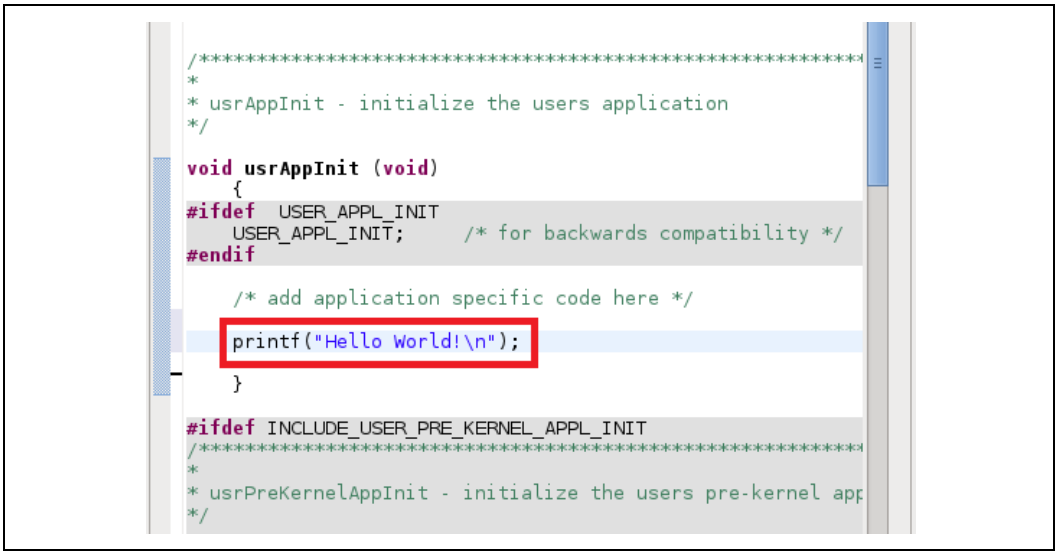
Press **Enter** twice. The cursor advances downward.

Type the following code:

```
printf("Hello World!\n");
```

See the boxed content in Figure 48 – Print File Code.

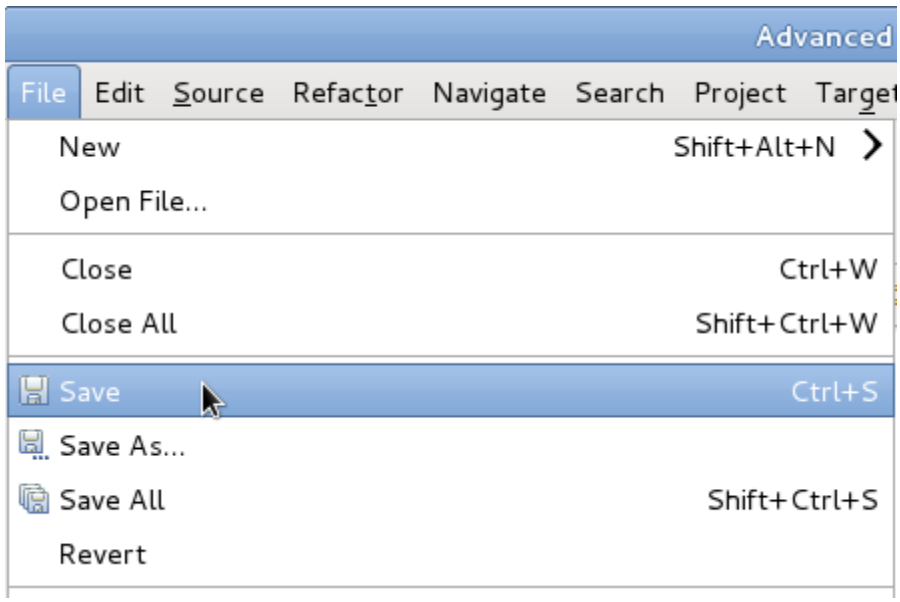
Figure 48 – Print File Code



You may optionally insert your own code in the space above into the VxWorks project.

Go to **File > Save**. See Figure 49 – File > Save.

Figure 49 – File > Save



The file saves to the system.

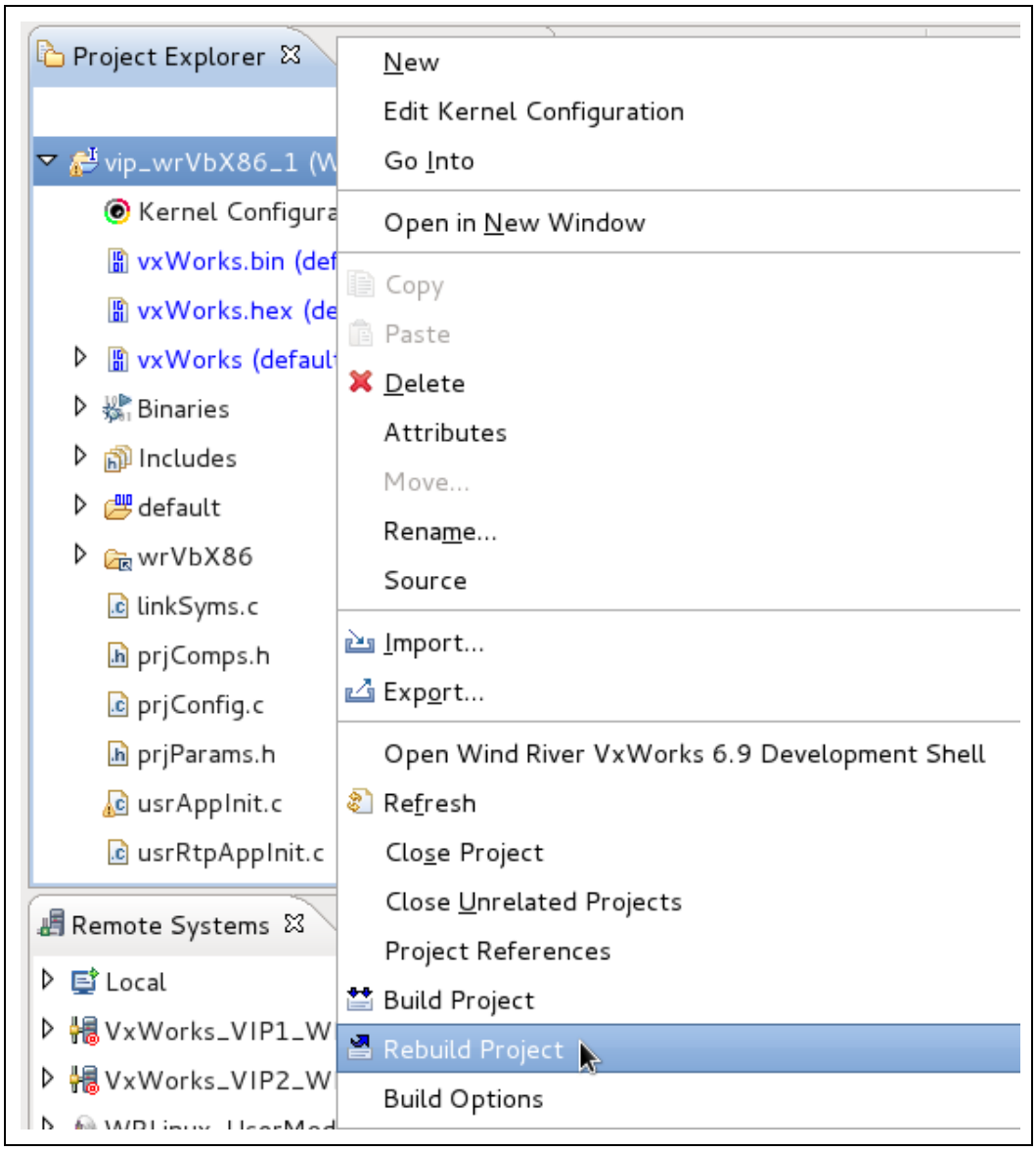
Repeat this procedure to embed source code for VxWorks #2 (**vip_wrVbX86_2**).

7.5.2 Build the Hypervisor Integration Project

In Project Explorer, go to and right-click the **vip_wrVbX86_1** object.

Select **Rebuild Project**. See Figure 50 – Rebuild Project.

Figure 50 – Rebuild Project



A **Rebuild Project – vip_wrVbX86_1** window appears (see Figure 51 – Rebuild Project) and a **Build Console** at the bottom of Workbench (see Figure 52 – Build Console) displays lines of scrolling code.

Figure 51 – Rebuild Project

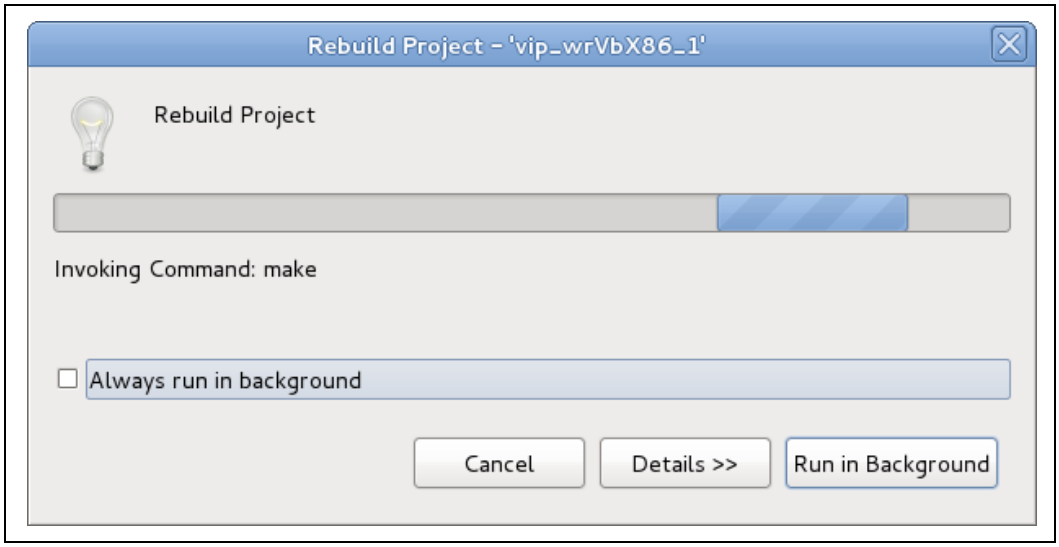
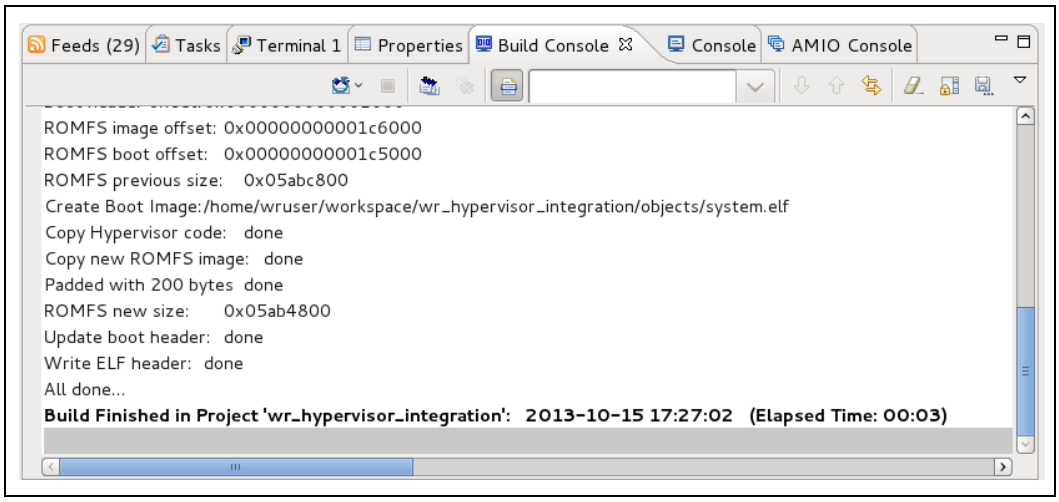


Figure 52 – Build Console



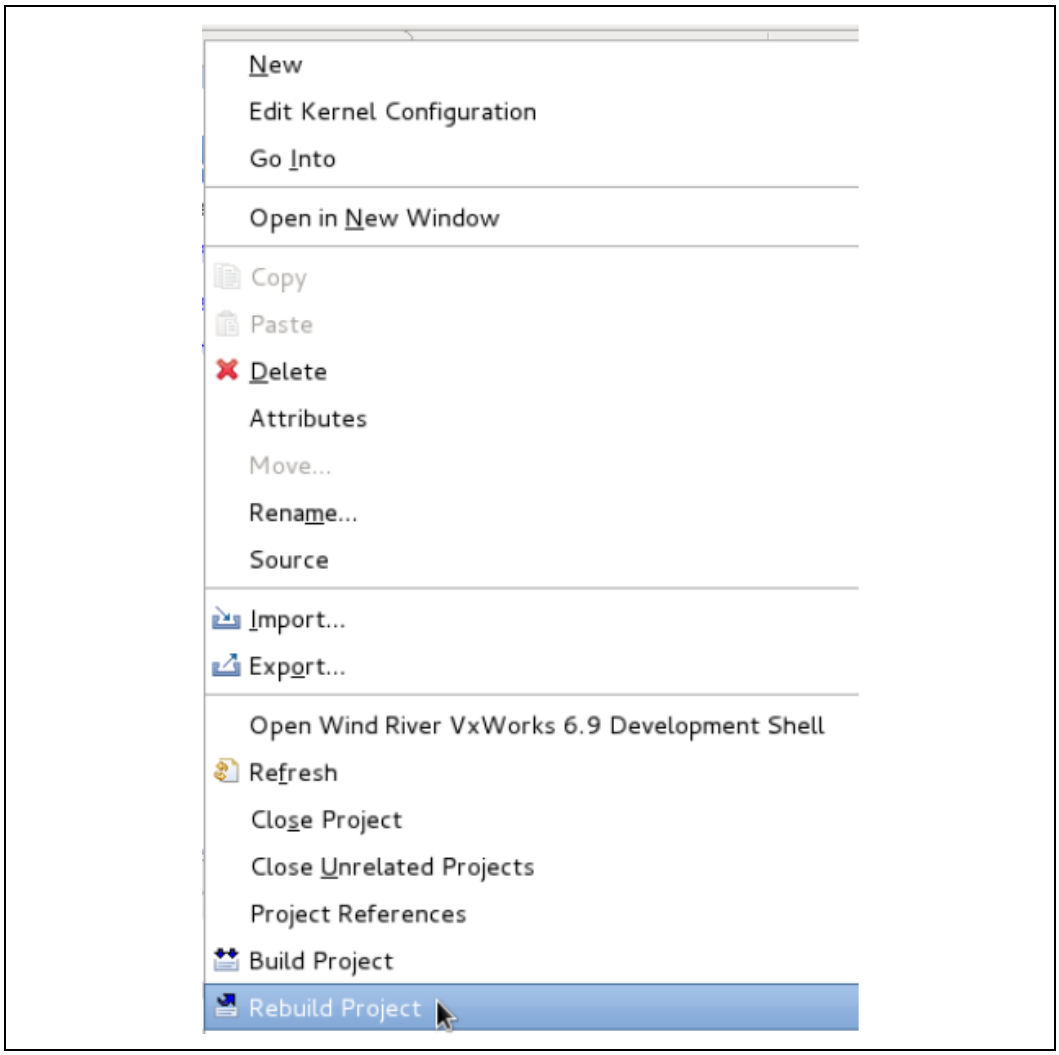
When the system finishes rebuilding the project, the **Rebuild Project – vip_wrVbX86_1** window disappears and returns to Wind River Workbench. Repeat this procedure for the VxWorks 2 (**vip_wrVbX86_2)** project**.**

In the procedure that follows, you instruct Hypervisor to collect the changes you just made in the VxWorks 1 project. As it collects these changes, Hypervisor also gathers any other changes you may have made in the VxWorks 2 (**vip_wrVbX86_2**) and Linux (**wrlinux_ugos_glib_std**) projects. All of the changes contribute to creating an image you can later use to boot the target.

Go to and right-click the **wr_hypervisor_integration** object.

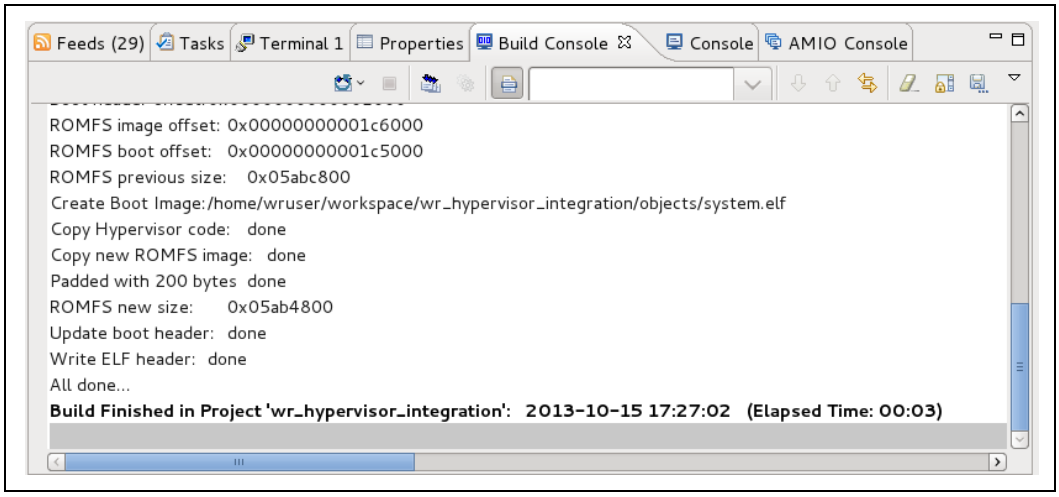
Select **Rebuild Project**. See Figure 53 – Select Rebuild Project.

Figure 53 – Select Rebuild Project



A **Rebuild Project – wr_hypervisor_integration** window appears and a **Build Console** pane at the bottom of Workbench displays lines of scrolling code. See Figure 54 - Build Console.

Figure 54 - Build Console



While the system rebuilds the **wr_hypervisor_integration** project, the system copies information from the other projects in Project Explorer. This process consolidates this information into a file called **system.elf**. The **system.elf** file comprises the image containing the modifications and configurations (including any code you create) that the target platform uses to boot.

Depending on the size of project you rebuild, after a few moments the **Rebuild Project** window disappears.

7.5.3 Copy the SYSTEM.ELF File to Boot Media

While there are several methods to move the **system.elf** file from the development host to the target platform, the easiest method is to copy the **system.elf** file to USB media and boot the target platform from that media.

Follow the steps to perform this copy procedure.

CAUTION: Intel Corporation provides you with the target platform media USB for copying the system.elf file from the development host to the target platform. However, copying data onto the target platform USB media may cause the data on the media to be overwritten. Make certain to back up data before using the media. Failure to do so may result in the loss of the data on the media.

IMPORTANT: Intel Corporation provides you with the *Target Platform* media USB for copying the **system.elf** file from the development host to the target platform. While you may use any Linux-formatted USB media, this procedure assumes you use the *Target Platform* USB media.

At the development host, insert the *Target platform* USB media into an open USB port. The USB media mounts to the Linux file system.

NOTE: You may optionally insert any Linux-formatted USB media (with 16 GB or more free space) into an open USB port.

Open the the *Target platform* USB media using the development host Linux File Explorer.

Open the **/images** directory and find the **system.elf** file.

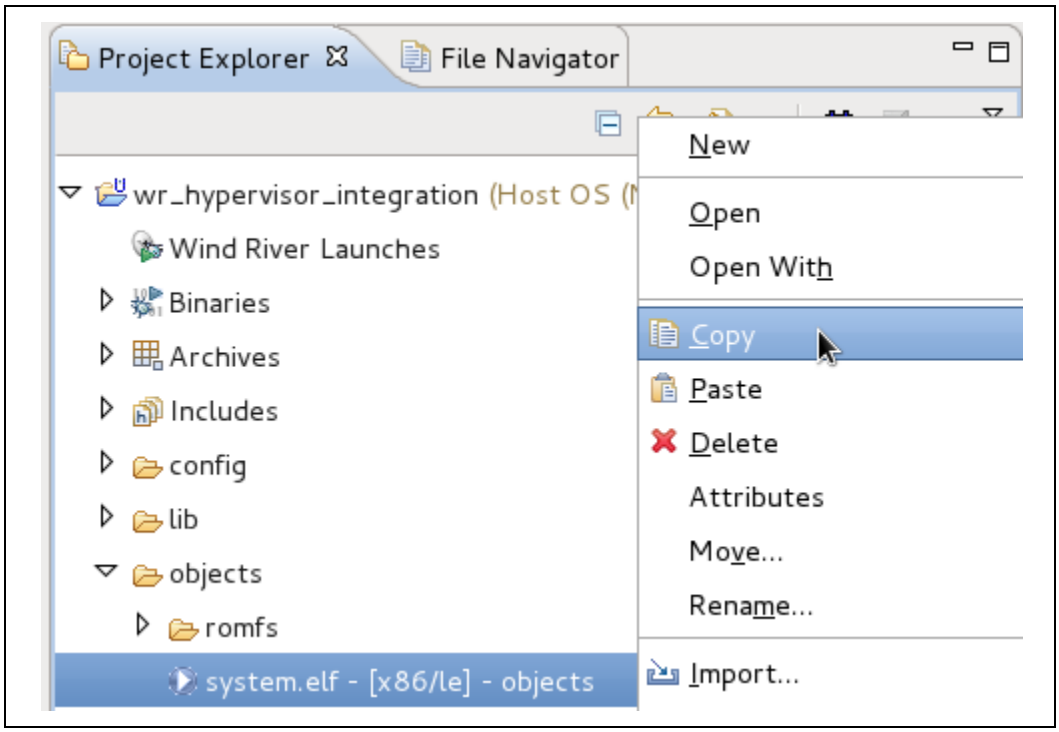
Rename the **system.elf** file to **system.elf.factory**. The file you rename in this step is a default factory-configured **system.elf** file which you may later wish to use to boot the target platform. By renaming it, you are protecting it from being overwritten in the following steps.

In Workbench Project Explorer, select the **wr_hypervisor_integration** project and expand it.

Expand the **Objects** folder.

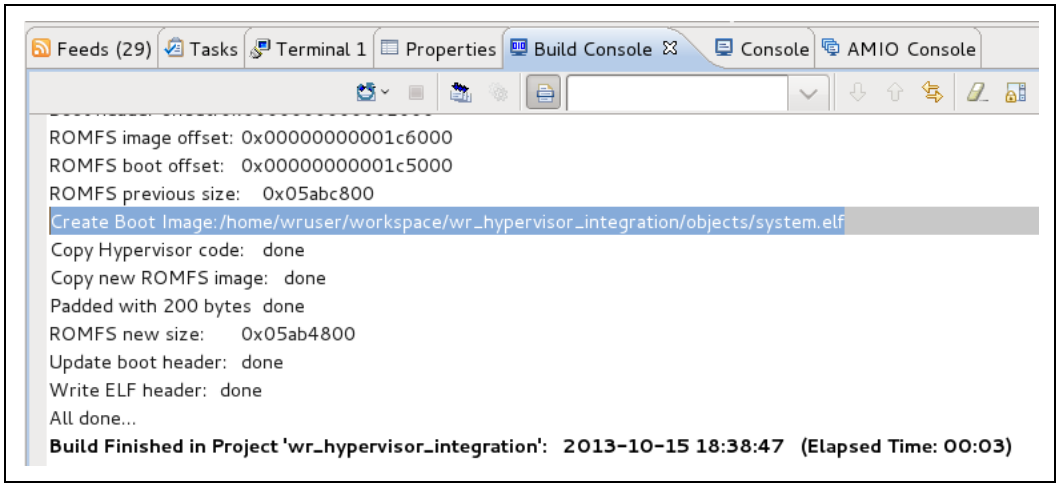
Right-click the **system.elf** object and select **Copy**. See Figure 55 – System.elf Copy.

Figure 55 – System.elf Copy



You may optionally use a command line or the Linux file explorer to copy the **system.elf** file from the location indicated in the **Build Console**. For a path to the **system.elf** file, see the darkened content in Figure 56 – System.elf Directory.

Figure 56 – System.elf Directory



In Workbench, close the AMIO channel consoles.

Minimize Workbench.

Use Linux File Explorer to navigate to the following directory of the USB media:

```
/target_usb/images
```

Paste the **system.elf** file into the directory.

After the file finishes copying, unmount the USB media and remove it from the development host.

7.6 Configure Target Platform BIOS to Boot from USB Media

Before booting the target platform from USB media, you first configure the target platform BIOS to boot from USB media.

NOTE: To ensure that you start the target platform from USB media, follow this procedure each time you need to boot from USB media.

Shut down the target platform and wait 5 seconds.

Ensure that no other bootable media are inserted into the target platform.

Place the bootable USB media into an open USB port (preferably a blue USB 3.0 port).

NOTE: For a faster boot, Intel Corporation recommends you connect the boot media to a USB 3.0 port on the target platform. Arrows in Figure 57 - USB 3.0 Ports provide the locations of two target platform USB 3.0 ports.

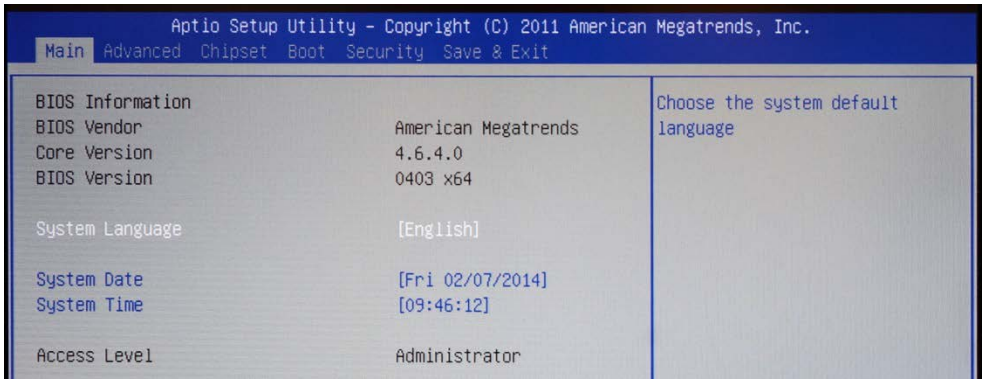
Figure 57 - USB 3.0 Ports

Press and release the target platform power button. The target platform begins to start.

As the target platform begins to start, repeatedly press the **DELETE** button.

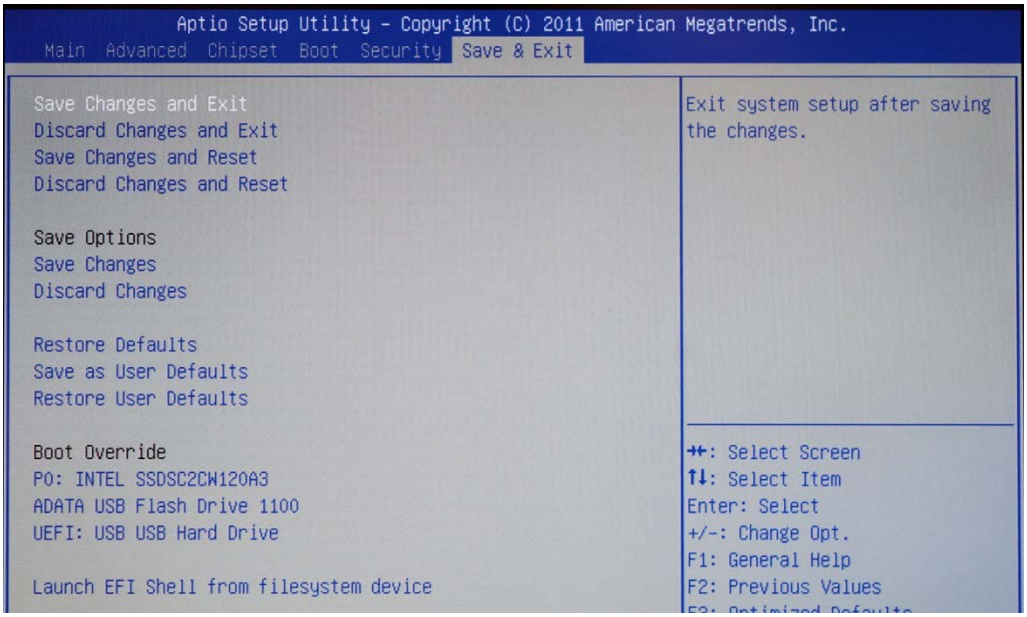
The target platform enters the BIOS **Main** tab. See Figure 58 – BIOS 1.

Figure 58 – BIOS 1


Aptio Setup Utility – Copyright (C) 2011 American Megatrends, Inc.
Main Advanced Chipset Boot Security Save & Exit
BIOS Information
BIOS Vendor American Megatrends
Core Version 4.6.4.0
BIOS Version 0403 x64
System Language [English]
System Date [Fri 02/07/2014]
System Time [09:46:12]
Access Level Administrator
Choose the system default language

Use the right-arrow key to go to the **Save & Exit** tab. See Figure 59 – BIOS 2.

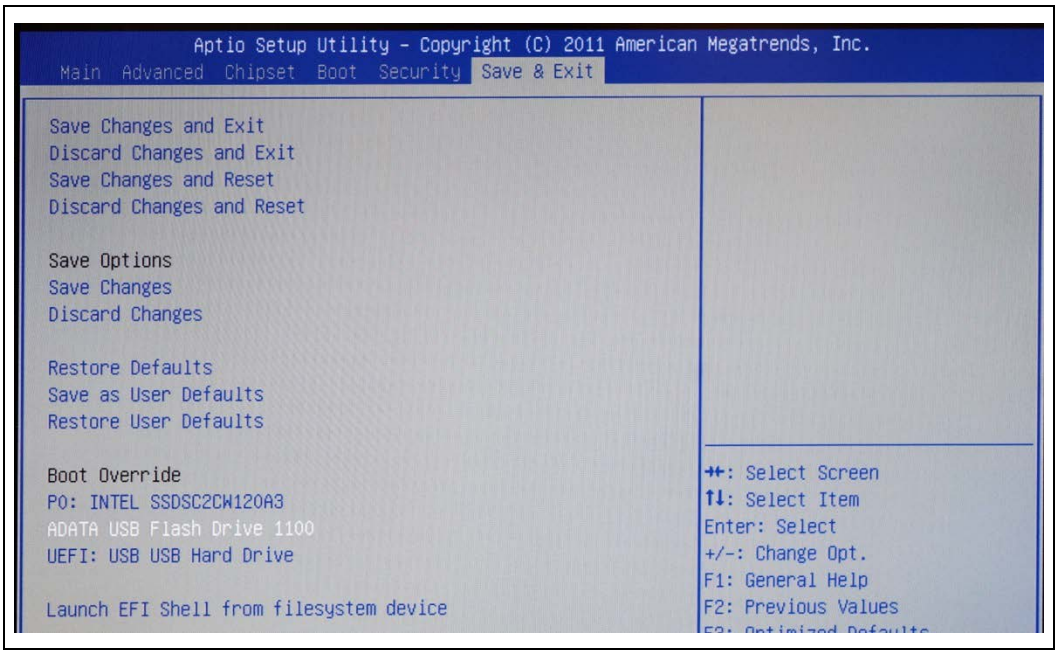
Figure 59 – BIOS 2



Use the down-arrow key to move the highlighted selection to the USB media. See Figure 60 – BIOS 3.

NOTE: In this example, the USB media is named **ADATA USB Flash Drive 1100**. Your USB media may have a different name.

Figure 60 – BIOS 3



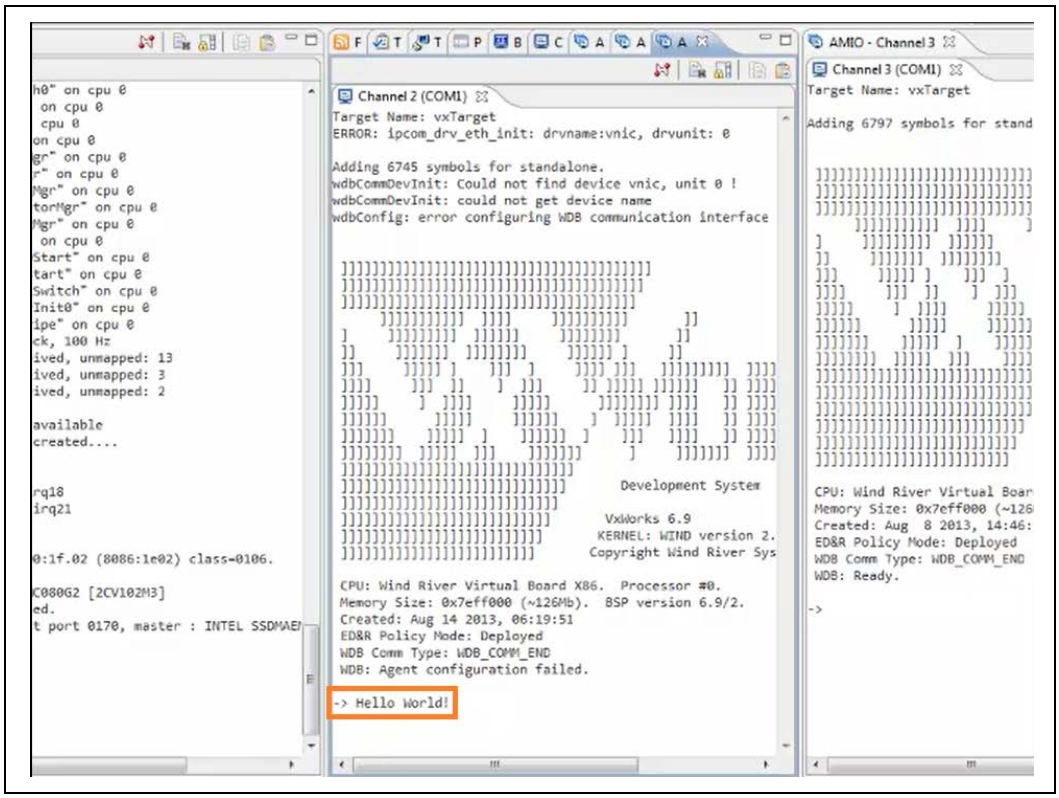
Press **Enter**. The target platform boots from USB media.

As the system starts, communication activity appears on the development host AMIO console. Several console tabs appear within the AMIO pane:

- **Core OS –** This shows the Hypervisor I/O
- **Channel 1 –**This shows the Linux I/O
- **Channel 2 –** This shows the I/O for VxWorks #1
- **Channel 3 –** This shows the I/O for VxWorks #2

The code you inserted into the **usrAppInit.c** object in the development host VxWorks project runs and displays. See the boxed content in Figure 61 – AMIO Consoles.

Figure 61 – AMIO Consoles


Channel 2 (COM1)
Target Name: vxTarget
ERROR: ipcom_drv_eth_init: drvname:vnic, drvunit: 0
Adding 6745 symbols for standalone.
wdbCommDevInit: Could not find device vnic, unit 0 !
wdbCommDevInit: could not get device name
wdbConfig: error configuring WDB communication interface
Development System
VxWorks 6.9
KERNEL: WIND version 2.
Copyright Wind River Sys
CPU: Wind River Virtual Board X86. Processor #0.
Memory Size: 0x7eff000 (~126Mb). BSP version 6.9/2.
Created: Aug 14 2013, 06:19:51
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Agent configuration failed.
-> Hello World!
AMIO - Channel 3
Channel 3 (COM1)
Target Name: vxTarget
Adding 6797 symbols for stand
CPU: Wind River Virtual Boar
Memory Size: 0x7eff000 (~126
Created: Aug 8 2013, 14:46:
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Ready.
->

This concludes a basic demonstration of working with preconfigured Workbench projects.

§

8 Workflow 3: Build Linux & VxWorks OS Images from Scratch

8.1 About this Workflow

This workflow is the long-form version of Workflow 2: Develop With Preloaded Workbench Projects, which uses prebuilt Project Explorer projects for quick productivity. Here you follow very detailed instructions to create projects (VSB, VIP, Linux Platform Project, Hypervisor Integration project) from scratch. Use these instructions, for example, if you determine that using the prebuilt projects that come with Workbench do not meet your development needs.

After building new projects from scratch, you can then use them to create a new target platform boot image. Once built, you then load the target image onto the target platform and boot it up. In this workflow you will have the opportunity to learn how to add your own source code to both Linux and the VxWorks VMs.

8.2 Preface

The Intel® Industrial Solutions System Consolidation Series (SCS) target platform system loads three guest operating systems (GOSes). Using Wind River Workbench, you can configure, build, and load three OSes onto Wind River Hypervisor. These operating systems are:

- One (1) instance of Wind River* Linux*
- Two (2) instances of Wind River* VxWorks*

Wind River VxWorks and Wind River Linux can be customized to suit your needs. For example, networking components like *telnetd* or *sshd* may be added to or removed from the OS configuration. Instructions for configuring components appear later.

Follow this procedure to create your own Wind River VxWorks, Linux, and Wind River Hypervisor images based on configurations readily available from the Workbench example repository. In this procedure, you configure and build images on the development host, and then load them to the target.

8.3 The VxWorks Source Build (VSB) Project

Workbench Project Explorer contains a VxWorks Source Build (VSB) project named **vsb_wrVbX86**. This project represents the pre-built libraries that make up vxWorks compiled specifically for the SCS virtual machines. There is no need to modify the core libraries of this VSB.

The steps in the next section will describe how to use the existing VSB to create your own VxWorks image. Proceed to the next step to create a VxWorks Image Project (VIP).

8.4 Create VxWorks Guest OS #1

SCS comes with two instances of Wind River* VxWorks OSes as virtual machines.

Follow these instructions to build VxWorks operating system #1.

NOTE: For further information about configuring and creating guest OSes, you may optionally see *Hypervisor User's Guide*, section 16.2: *Example: Hypervisor Standard Profile with a VxWorks Guest (Intel)*.

8.4.1 Create the VxWorks Image Project (VIP)

This procedure provides instructions to create a VxWorks image project.

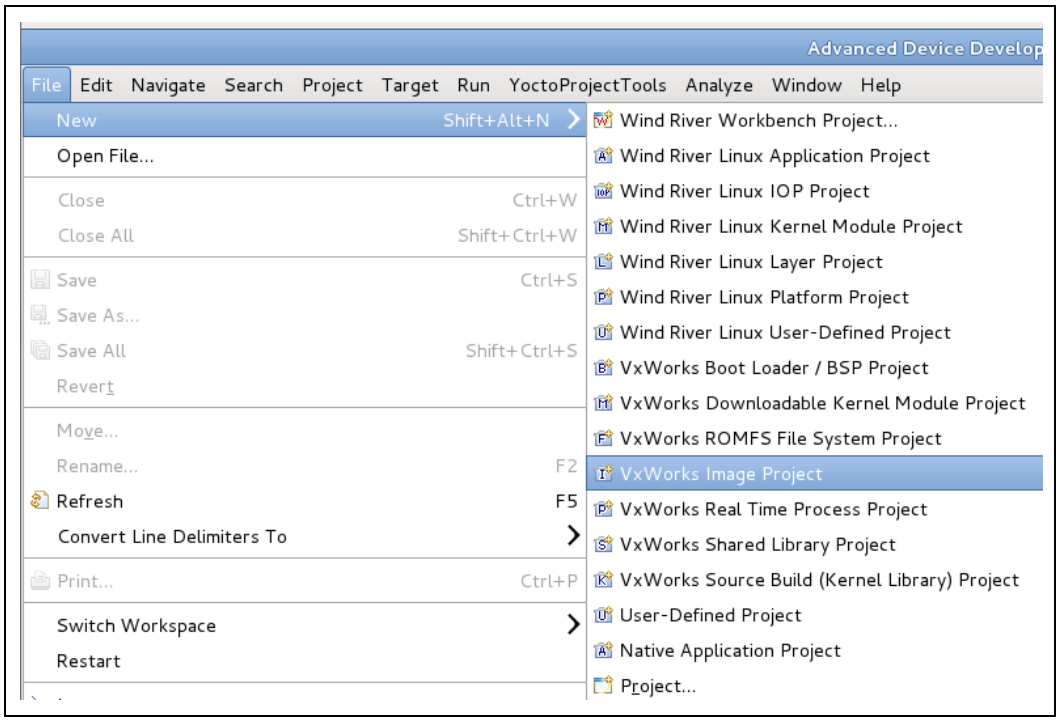
NOTE: For more information see *Hypervisor User's Guide, Section 16.3: Example: Hypervisor Standard Profile with a VxWorks Guest (Intel)*, a separate document that you may find by going to **Help** in Wind River Workbench.

Open Wind River Workbench.

Switch to the Advanced Device Development perspective: **Window > Open Perspective > Advanced Device Development**.

To open the VxWorks Image Project Wizard, go to **File > New > VxWorks Image Project.** See Figure 62 - File > New > VxWorks Image Project.

Figure 62 - File > New > VxWorks Image Project

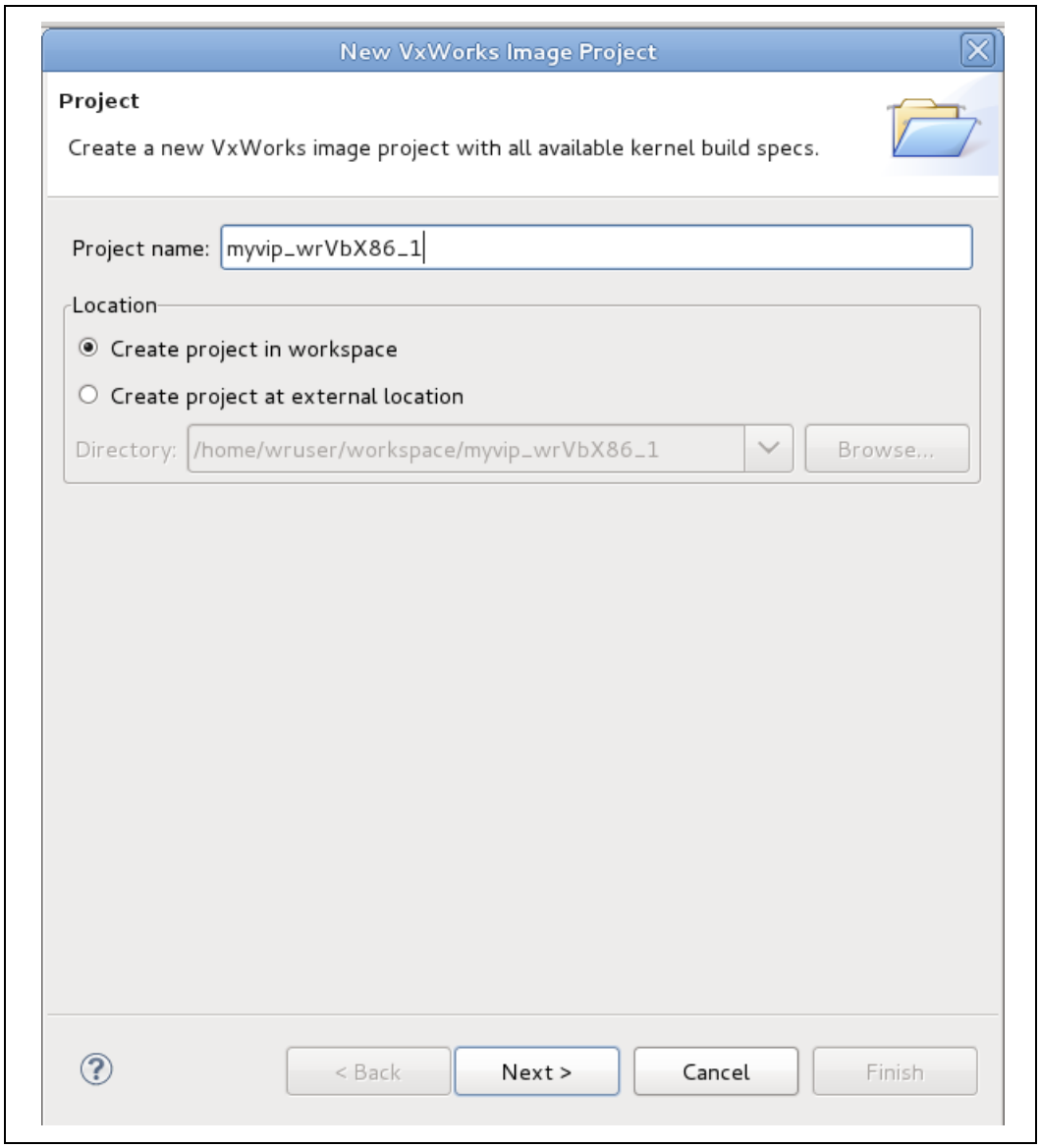


A New VxWorks Image **Project** window appears.

In the **Project name:** field, type **myvip_wrVbX86_1**

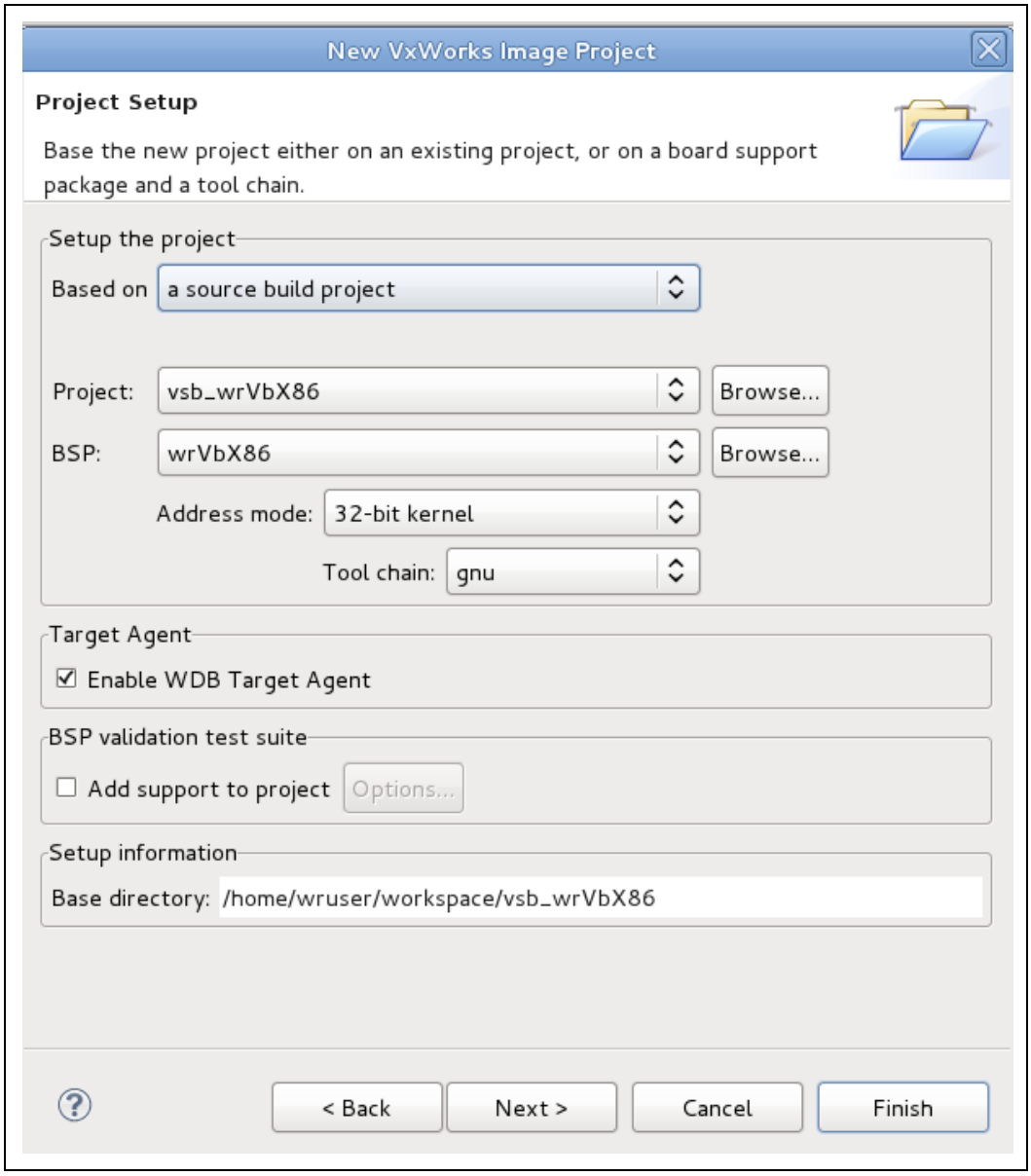
Select the **Create project in workspace** button. See Figure 63 - Create Project in Workspace.

Figure 63 - Create Project in Workspace



Click **Next**. The **Project Setup** window appears. See Figure 64 - vsb_wrVbX86.

Figure 64 - vsb_wrVbX86



At the **Based on** drop-down menu, select **a source build project**.

From the **Project** drop-down menu select **vsb_wrVbX86**. See Figure 64 - vsb_wrVbX86.

Check the **Enable WDB Target Agent** checkbox.

To create the VIP project, click **Finish**.

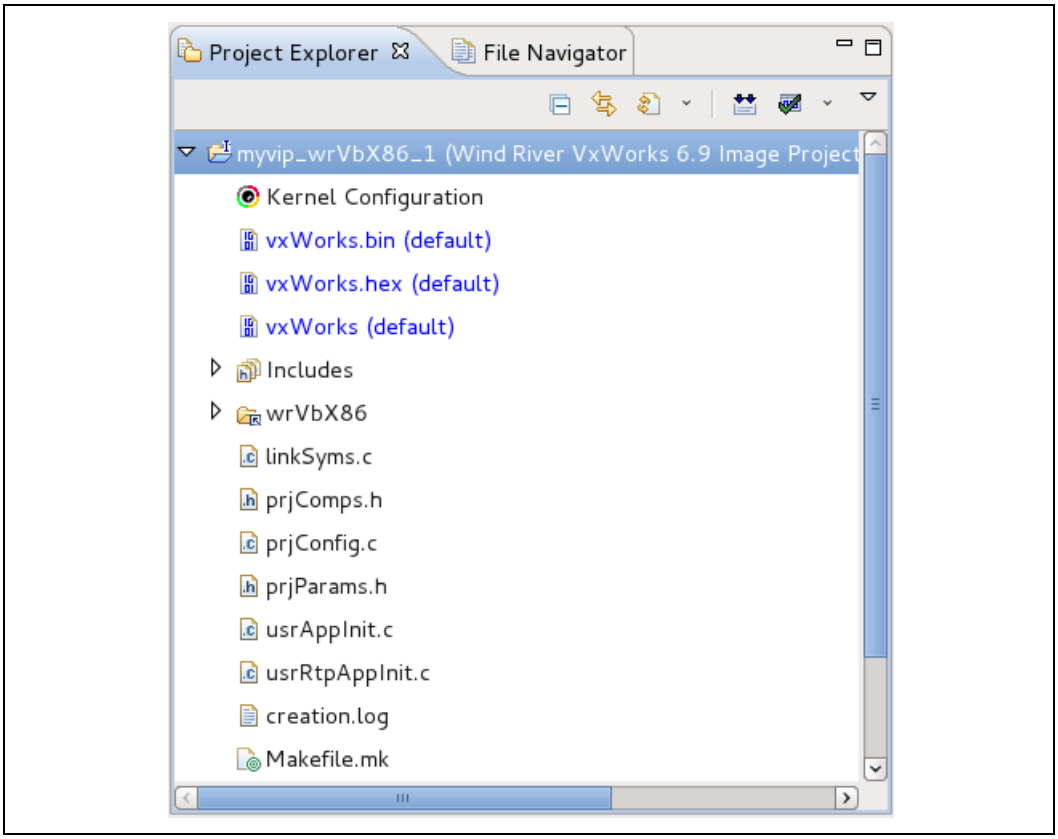
After a few moments, a **myvip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)** appears in the **Project Explorer** pane.

Go to **Project Explorer**.

Double-click the vip project object you just created. In this instruction, the object is named **myvip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)**.

Project details appear below the image project line. See Figure 65 - Project Details.

Figure 65 - Project Details



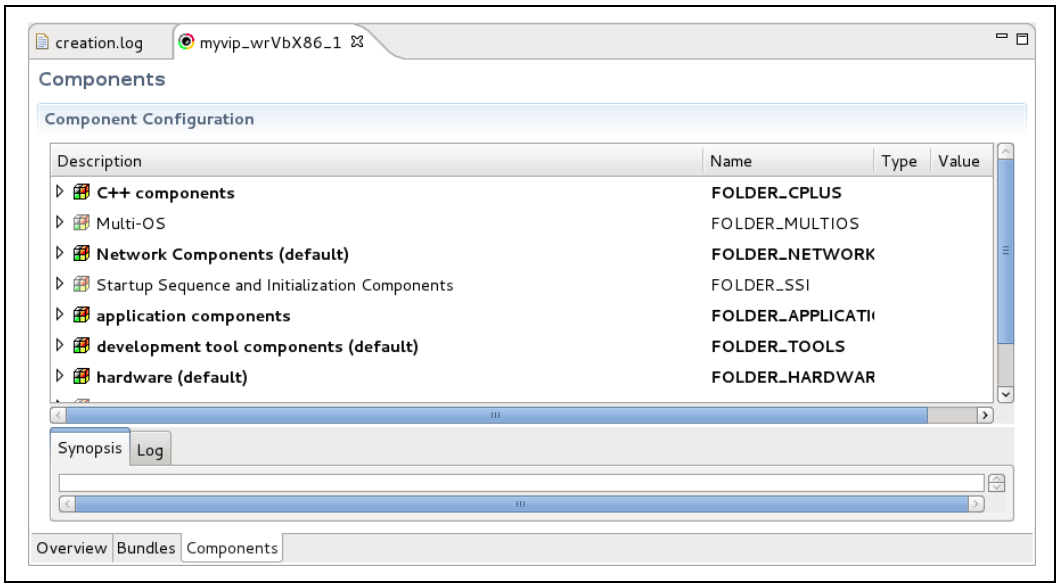
NOTE: If at this point, you do not wish to add components or applications to your VxWorks image project, proceed to Section 8.4.5 - Build the Project.

8.4.2 Add Components to VxWorks

VxWorks provides control over what components and capabilities are provided to the applications. The Kernel configuration is where you select the components. In a new project, components geared towards networking and debugging are minimal. This section includes some of these useful components.

Double-click the **Kernel Configuration** node. A **myvip_wrVbX86_1** components tab appears in the editor pane. See Figure 66 - Components Tab.

Figure 66 - Components Tab



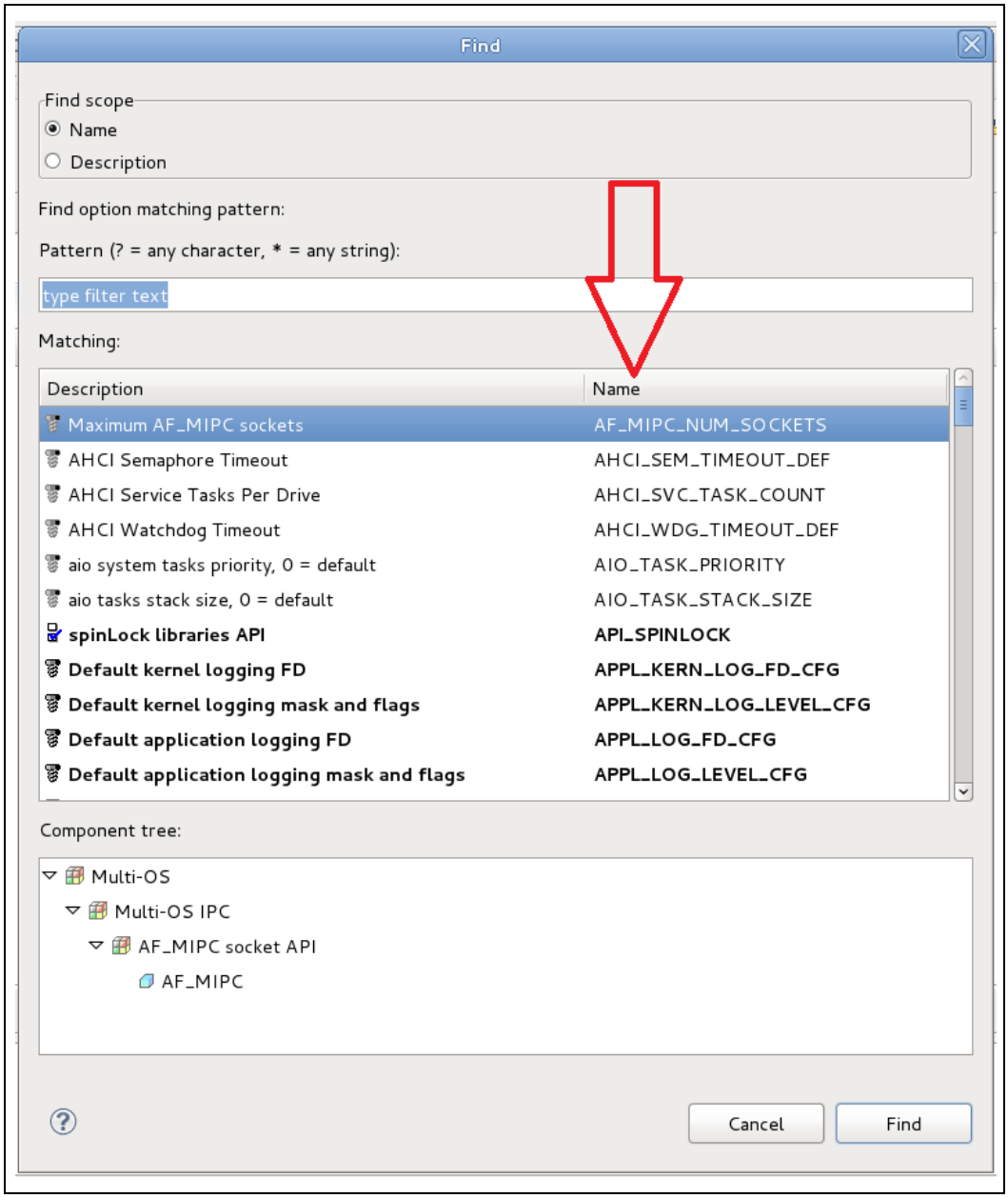
Double-click the **myvip_wrVbX86_1** tab to maximize the pane. Optionally double-click the tab again to restore the common Workbench workspace perspective.

In the following steps you add components to the **myvip_wrVbX86_1** project. Review the steps below and then perform them to add components in the optional and required components lists that appear later in this procedure.

Click the **myvip_wrVbX86_1** tab.

Press **Ctrl+F.** A **Find** window appears. See Figure 67 - Find Window.

Figure 67 - Find Window



The **Find** window allows you to search for components in the Kernel Configuration Tool.

Review the components listed in Table 4 – VIP INLCUDE Components.

Table 4 – VIP INLCUDE Components

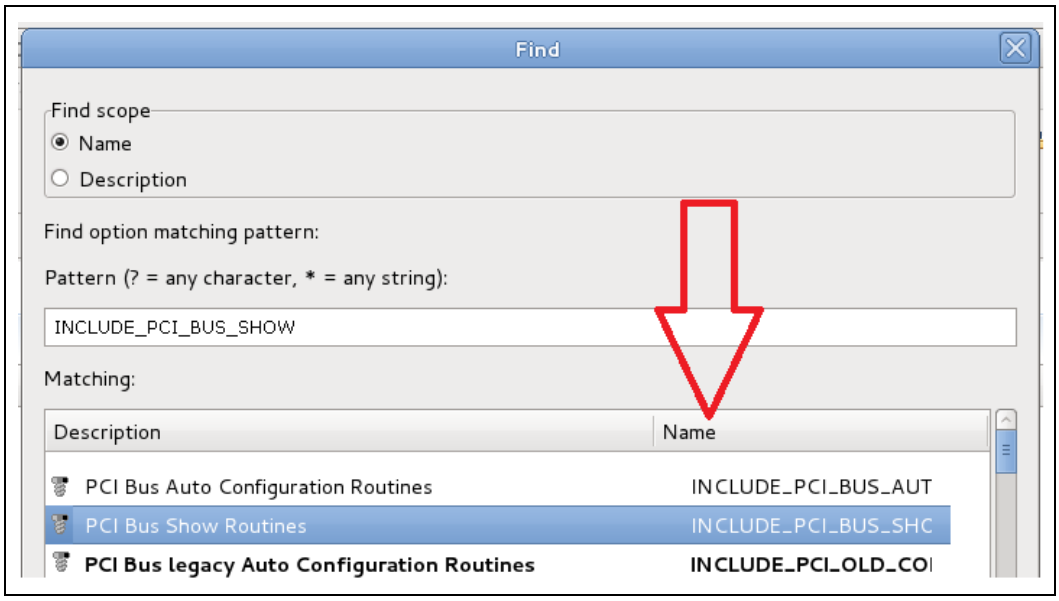
Optional Components	Required Components
INCLUDE_SHELL_BANNER	INCLUDE_USB_INIT
INCLUDE_IFCONFIG	INCLUDE_USB_MHDRC_OCD_HCD_INIT
INCLUDE_PING	INCLUDE_USB_MHDRC_OCD_TCD_INIT
INCLUDE_IPTELNETS	INCLUDE_USB_GEN2_STORAGE_INIT
INCLUDE_TELNET_CLIENT	INCLUDE_DOSFS_CACHE
INCLUDE_IPATTACH	INCLUDE_XBD_RAMDRV
INCLUDE_ROUTECMD	INCLUDE_XBD_PART_LIB
INCLUDE_WDB_ALWAYS_ENABLED	INCLUDE_HRFS
INCLUDE_WDB_PROXY	INCLUDE_DISK_UTIL
INCLUDE_ROMFS	DRV_SIO_NS16550
INCLUDE_SHOW_ROUTINES	
INCLUDE_TLS	
INCLUDE_CPLUS	
INCLUDE_CPLUS_LANG	
INCLUDE_ISR_OBJECTS	
INCLUDE_POSIX_MQ	
INCLUDE_POSIX_SEM	
INCLUDE_POSIX_PTHREADS	
INCLUDE_SYSCTL_CLI	
INCLUDE_TRIGGER_SHOW	
INCLUDE_WDB_DPRINTF	
INCLUDE_SHARED_DATA	
INCLUDE_IPDHCPC	
INCLUDE_PCI_BUS_SHOW	
INCLUDE_VXBUS_SHOW	
INCLUDE_VNIC_VXB_L2SWITCH_END	

Inside the **Find** window, in the **Pattern** field, type a component from the Optional list shown in Table 4 – VIP INLCUDE Components.

NOTE: This procedure uses the component named **INCLUDE_PCI_BUS_SHOW** as an example. This component is bolded in the Optional column in Table 4 – VIP INLCUDE Components.

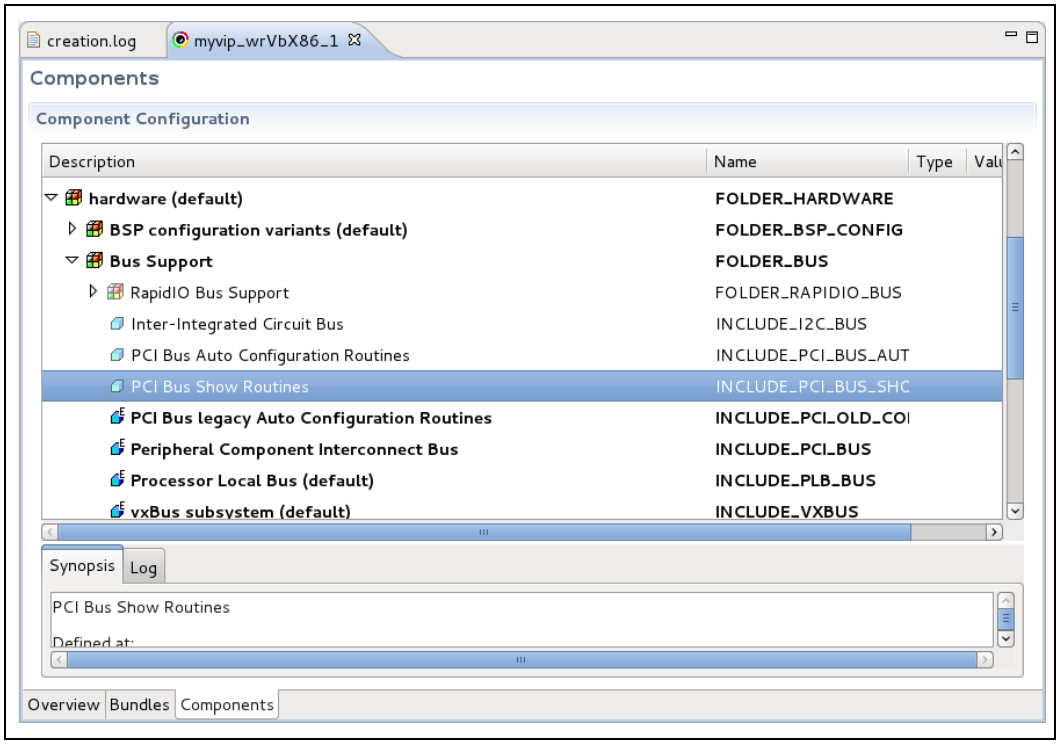
After typing the component name, look for the matching component name under the **Name** column. See the arrowed column in Figure 68 - Find Window 2.

Figure 68 - Find Window 2



Double click the matching object. The **Find** window closes and the component appears as a highlighted item in the **Component Configuration** tab. See Figure 69 - Highlighted Item.

Figure 69 - Highlighted Item



Right-click the component and select **Include (quick include)**. A progress window appears momentarily as the component loads. The system returns to the **Component Configuration** tab and the component you just added appears as a bolded item. See Figure 70 - Bolded Item.

Figure 70 - Bolded Item

Repeat the procedure to add the optional components from the **Optional Components** column in Table 4 – VIP INLCUDE Components.

Repeat the procedure to add each of the required components from the **Required Components** column in Table 4 – VIP INLCUDE Components.

When finished adding components, press **CTRL+S** to save the changes you have made.

8.4.3 Add User Applications to the VIP Using DKM

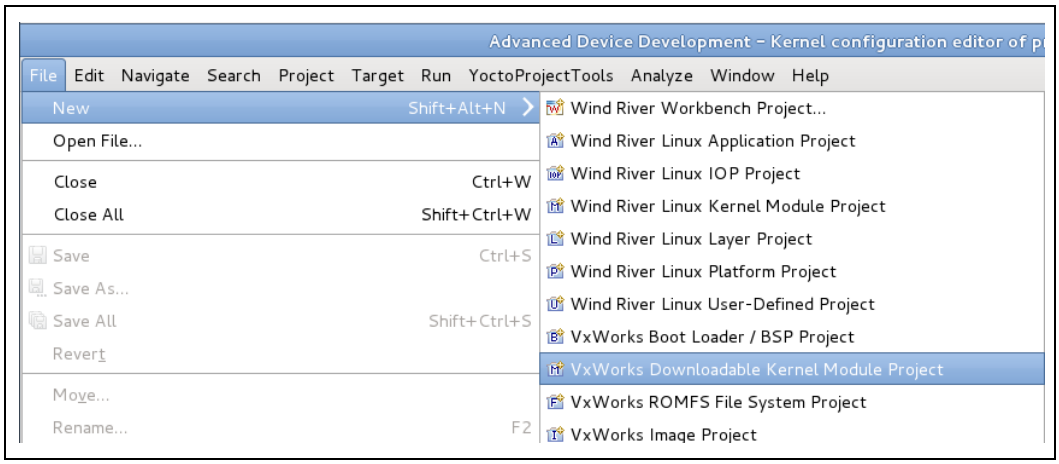
You can use a downloadable kernel module (DKM) project to manage and build modules that reside in the VxWorks kernel space. Workbench provides pre-configured templates for various types of projects, and lets you create and import these project templates.

In this workflow, when your DKM development work is complete, you link the DKM into the kernel. The following sections provide instructions to link your application into the kernel.

To create a DKM project, perform the following procedure:

Select **File>New>VxWorks Downloadable Kernel Module Project**. A window appears. See Figure 71 – New DKM.

Figure 71 – New DKM

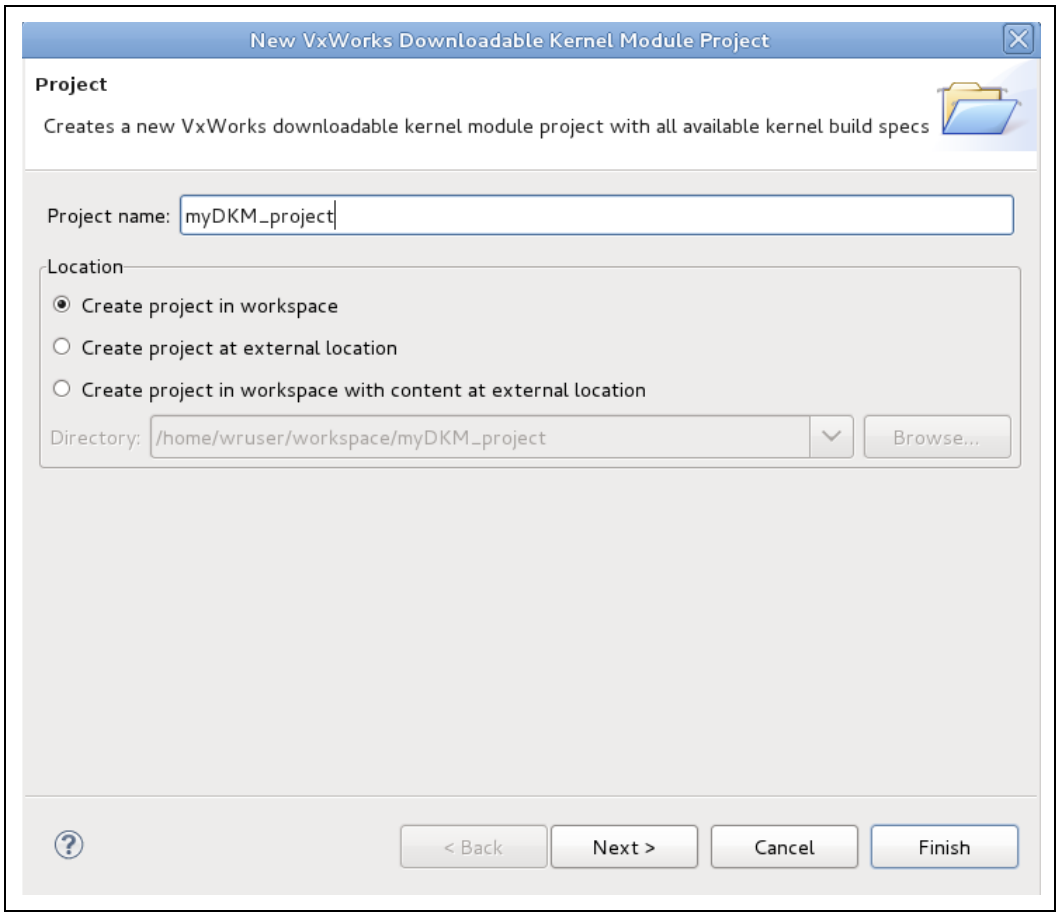


Enter a name for your project.

Select the **Create project in workspace** option. See Figure 72 - Create....

NOTE: When you create a new project using the **Create project in workspace** option, Workbench creates a directory bearing the same name as the project you choose in workbench's current workspace directory. See Figure 72 - Create....

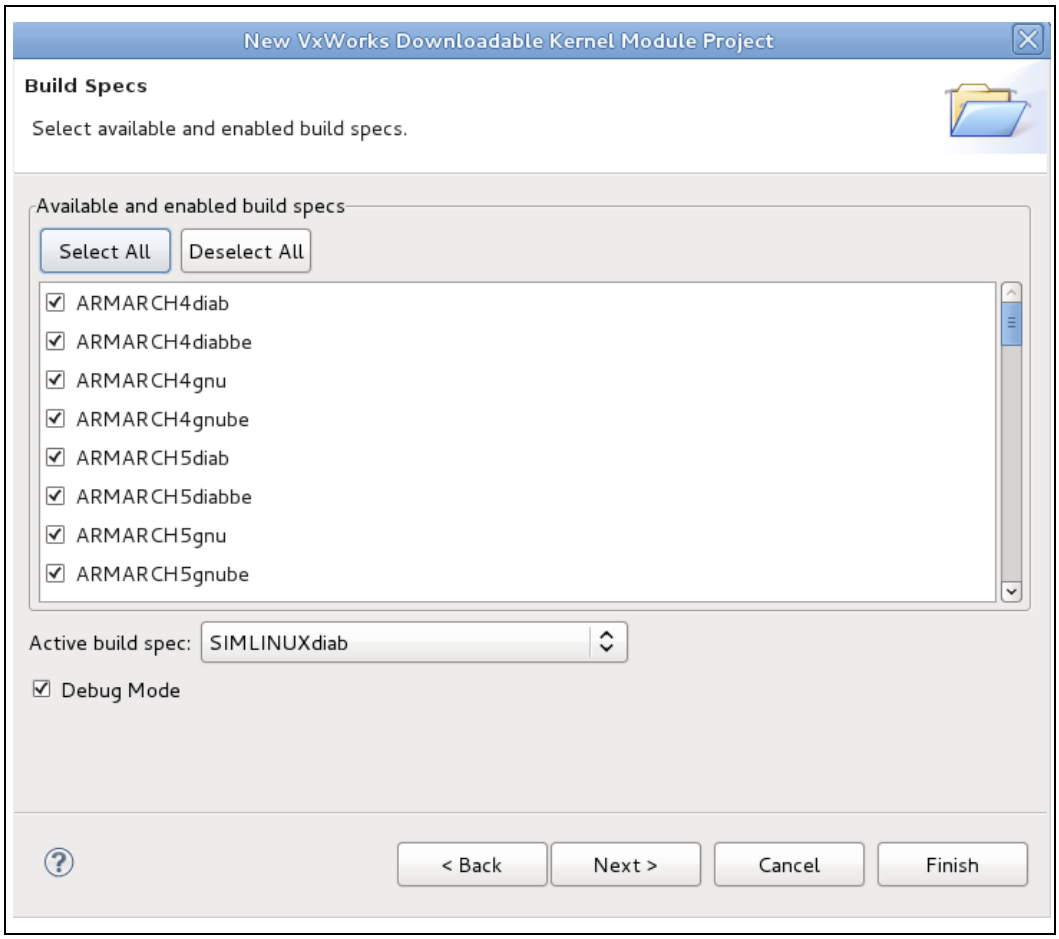
Figure 72 - Create...


New VxWorks Downloadable Kernel Module Project
Project
Creates a new VxWorks downloadable kernel module project with all available kernel build specs
Project name: myDKM_project
Location
Create project in workspace
Create project at external location
Create project in workspace with content at external location
Directory: /home/wruser/workspace/myDKM_project
Browse...
< Back
Next >
Cancel
Finish

Click **Next**.

Click **Next** several times until the **Build Specs** window appears. See Figure 73 - Build Specs Window.

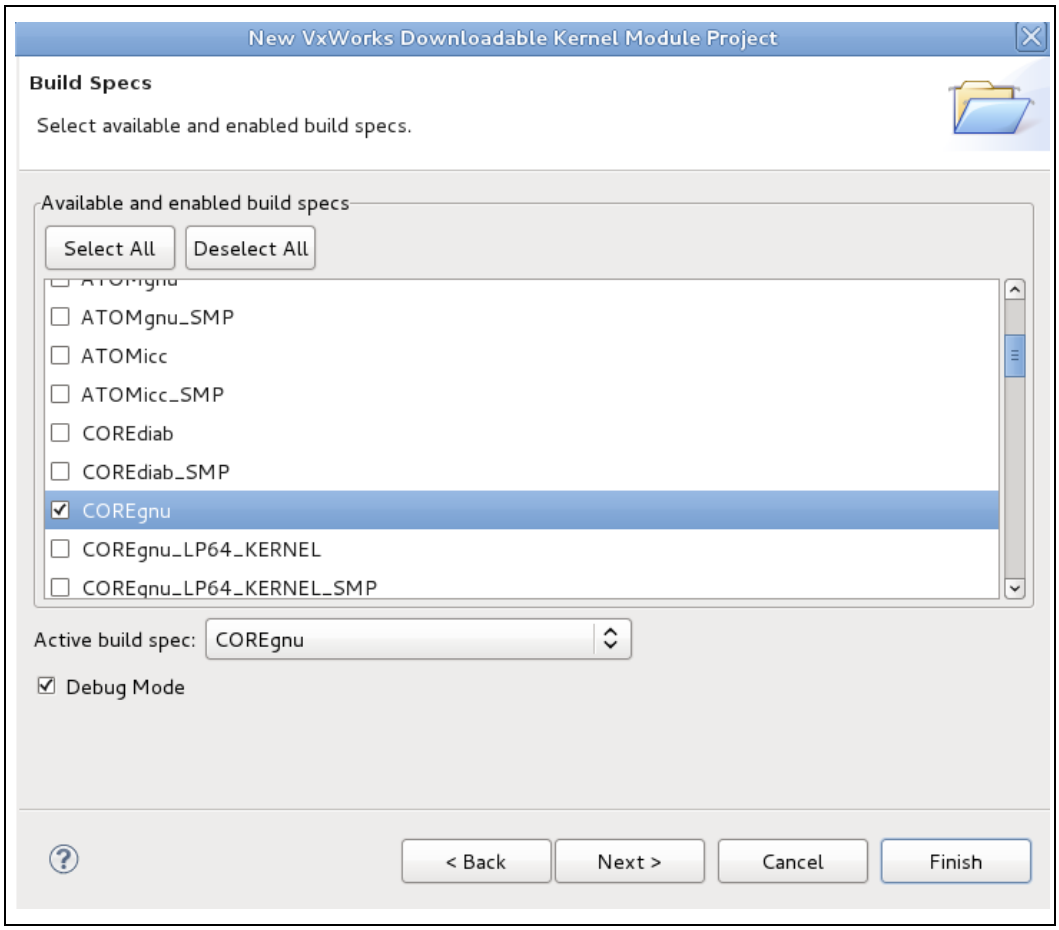
Figure 73 - Build Specs Window



In the **Build Specs** window, click **Deselect All**.

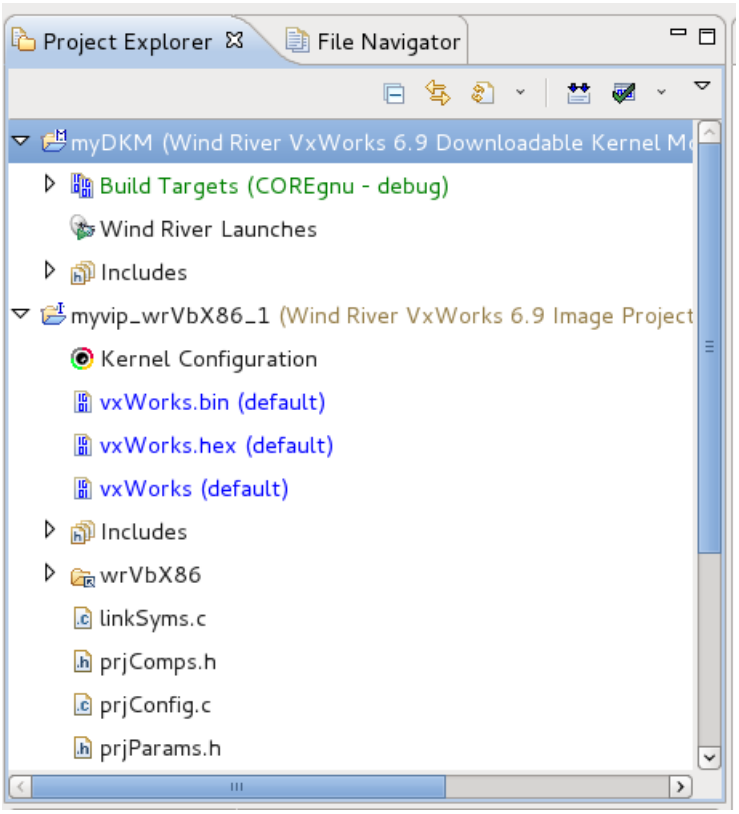
Scroll down the **Available and enabled build specs** list and select **COREgnu**. See Figure 74 – COREgnu.

Figure 74 – COREgnu



Click **Finish**. A **Wind River VxWorks Downloadable Kernel Module Project** line item appears in the Workbench Project Explorer window. See Figure 75 - DKM Module.

Figure 75 - DKM Module



8.4.4 Add Source Code to the Project

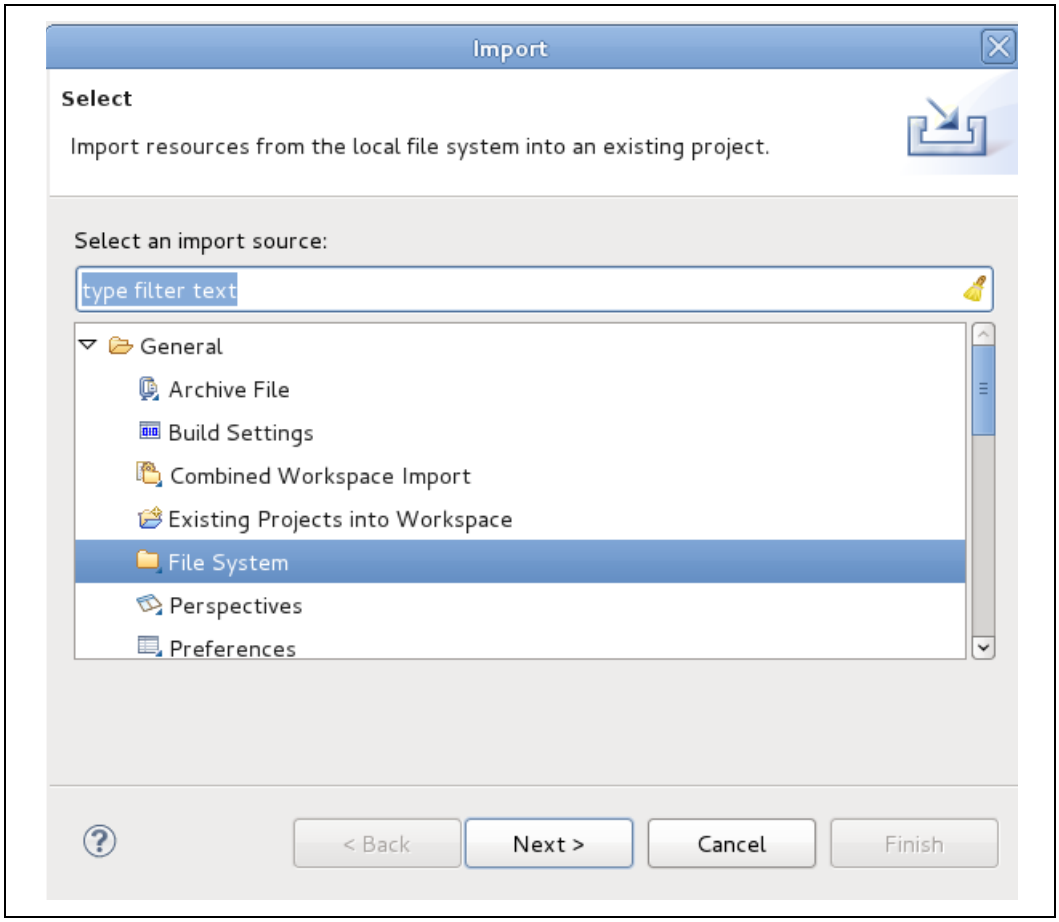
By this point you have set up the basic infrastructure to support your VxWorks project, but you have imported no application code to operate on this OS. In this procedure, you can import source files containing application code.

In **Project Explorer** right-click the **Downloadable Kernel Module Project** you just created and select **Import**. A **Select** window appears.

In the **Select an import source** list, click the triangle adjacent to **General**. An indented options list appears below **General**.

Select **File System**. See Figure 76 – File System.

Figure 76 – File System



Click **Next**. A **File system** window opens.

At the right of the **From directory:** field, click **Browse**. A **Recently Used** window appears.

Click **File System**.

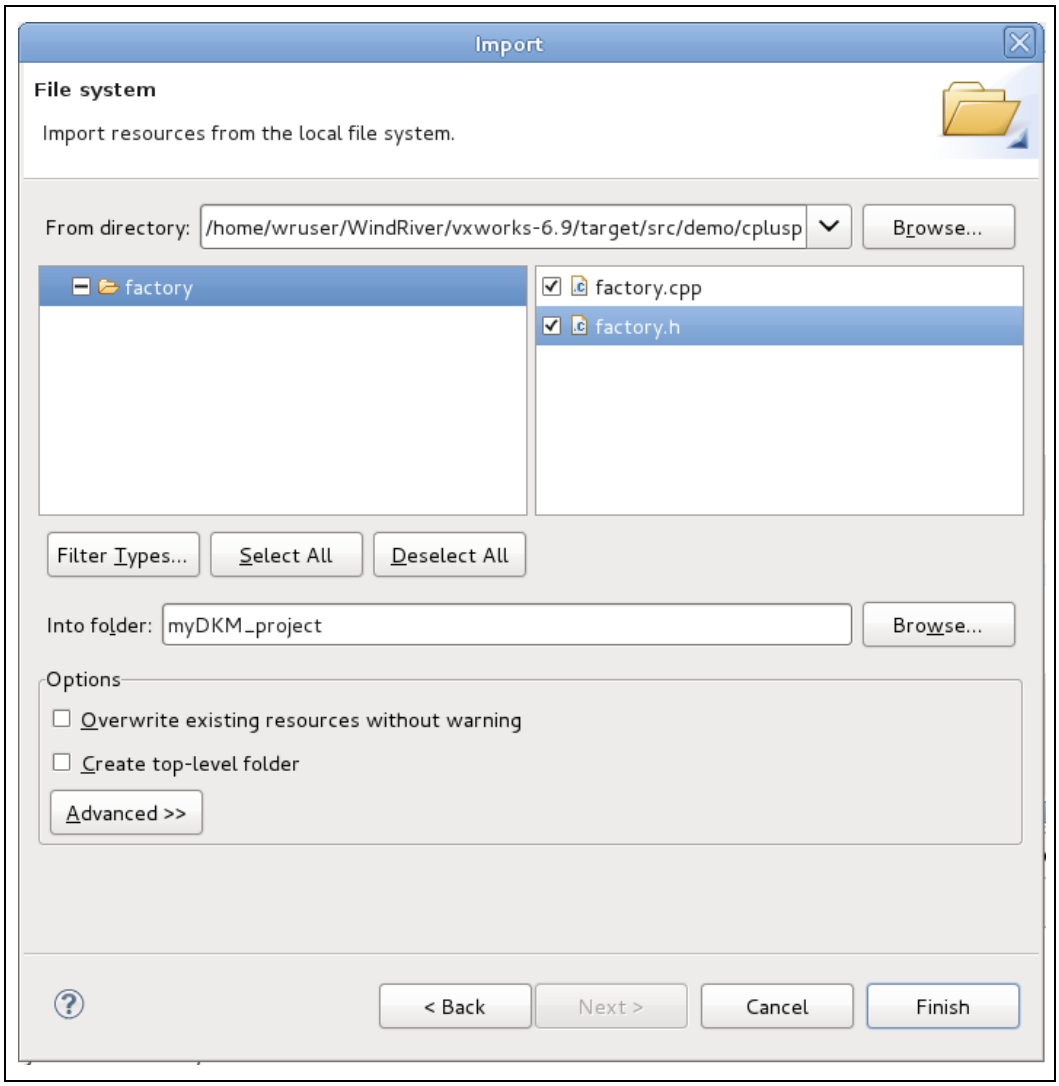
Browse to the directory that contains your source code.

NOTE: For demonstration purposes, you may optionally navigate to the following directory:

`/home/wruser/WindRiver/vxworks-6.9/target/src/demo/cplusplus/factory/`

Click **OK**. Your application source tree appears in the **Import** window. See Figure 77 – File System.

Figure 77 – File System



Under the right pane, check the application components you wish to enable.

Click **Finish**. The application source imports into Workbench.

8.4.5 Build the Project

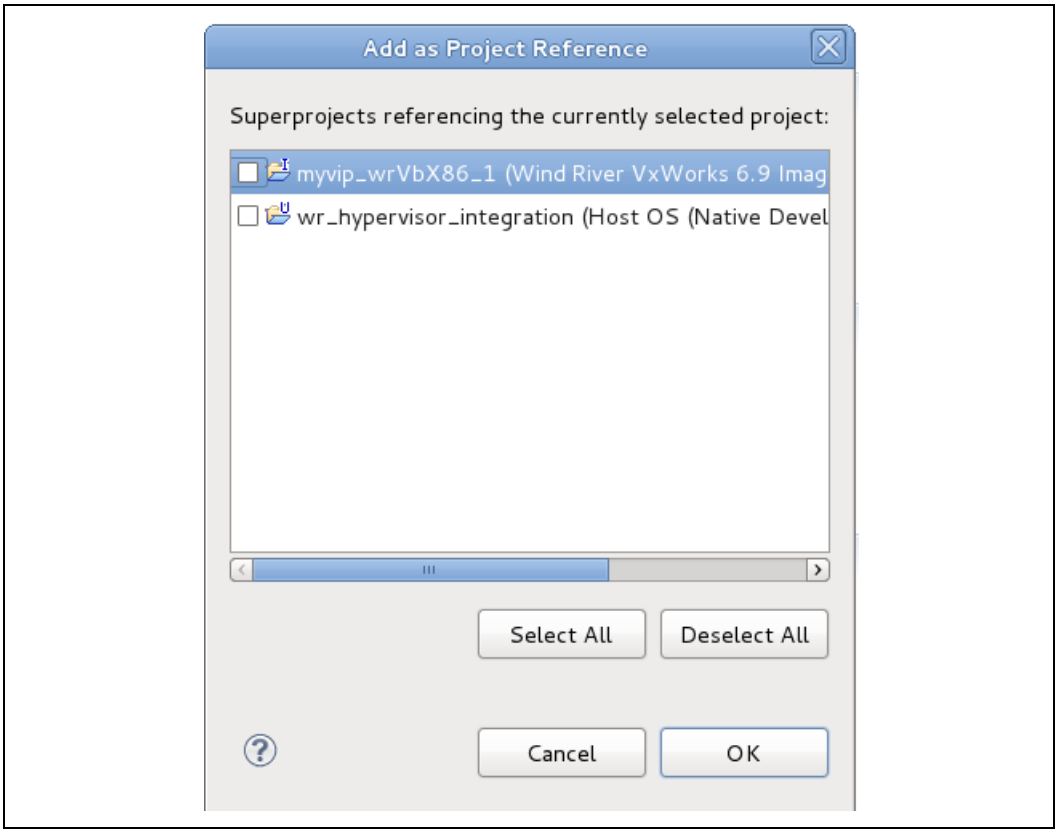
After you have created and configured your project, you must build it. You can use kernel applications linked with the VxWorks kernel itself. You can also create custom static libraries for use with kernel applications.

NOTE: For more information on projects and project structure, see *Wind River Workbench User's Guide: Projects Overview*.

You can build VxWorks kernel applications and associated libraries by adding your DKM project to the VIP and then build it. To add a DKM project to the VIP, follow this procedure:

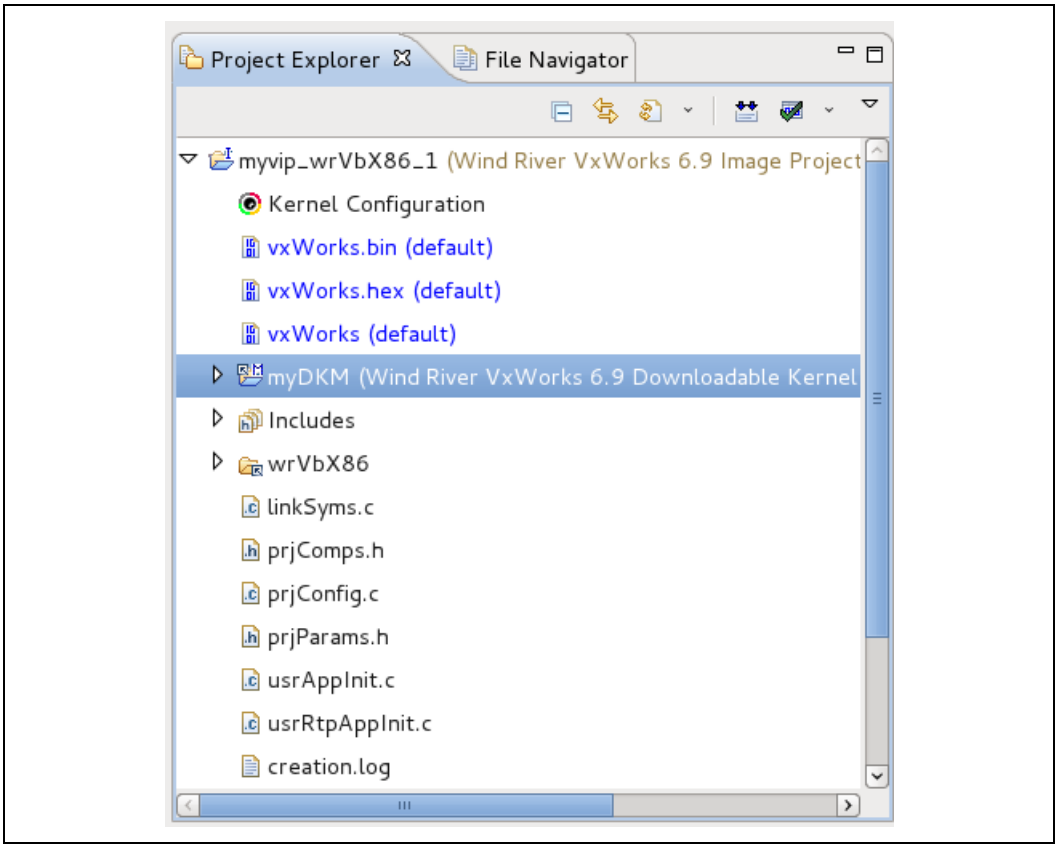
Right-click the Downloadable Kernel Module project, select **Project References > Add as Project Reference**. An **Add as a Project Reference** window appears. See Figure 78 – Project Reference.

Figure 78 – Project Reference



Select **myvip_wrVbX86_1** and click **OK**. In **Project Explorer**, the Downloadable Kernel Module project object appears within the **myvip_wrVbX86_1** project. See Figure 79 – Kernel Object.

Figure 79 – Kernel Object

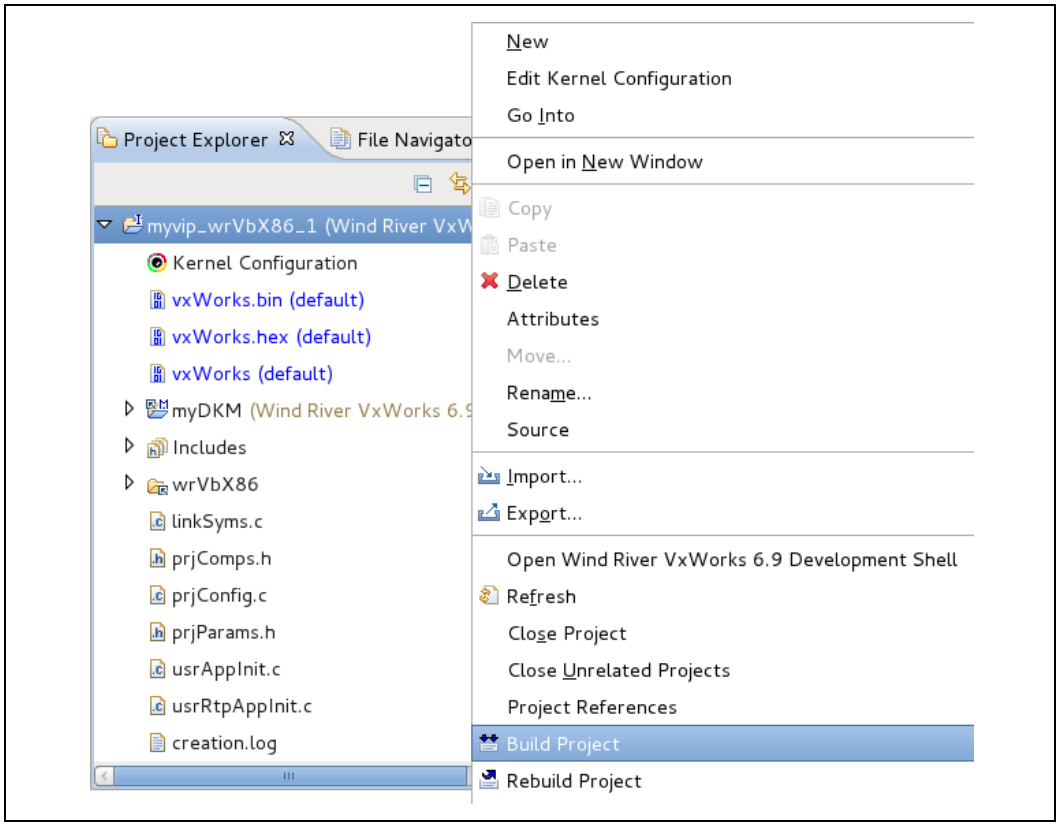


8.4.6 Build the VIP

To build the VxWorks Image Project (VIP), go to the **Project Explorer** pane and right-click **myvip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)**. A drop-down menu appears.

Select **Build Project**. See Figure 80 – Build Project.

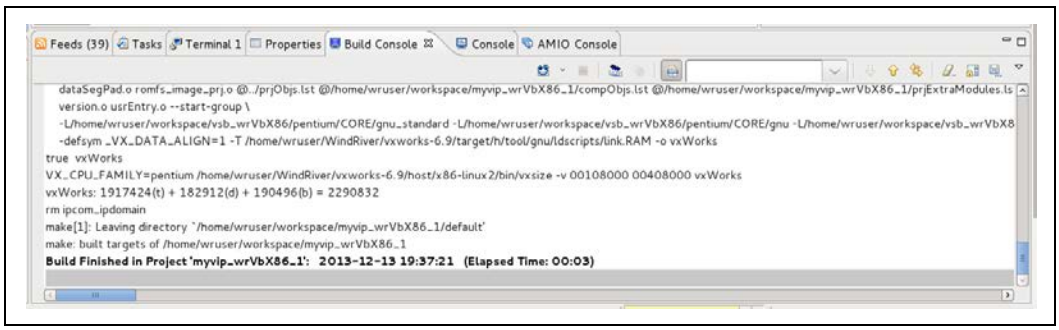
Figure 80 – Build Project



The build window shows lines of code compiling. A build project progress window appears, and a progress bar moves repeatedly left and right while the system compiles the VxWorks libraries required for the Guest OS.

When the project finishes building, a **Build Finished in Project vip_wrVbX86_1** statement appears at the bottom of the **Build Console** pane. See Figure 81 – Build Console.

Figure 81 – Build Console



The resulting executable is located in the project's default folder and is called **vxWorks**.

```
/home/wruser/workspace/myvip_wrVbX86_1/default
```

8.5 Create VxWorks Guest OS #2

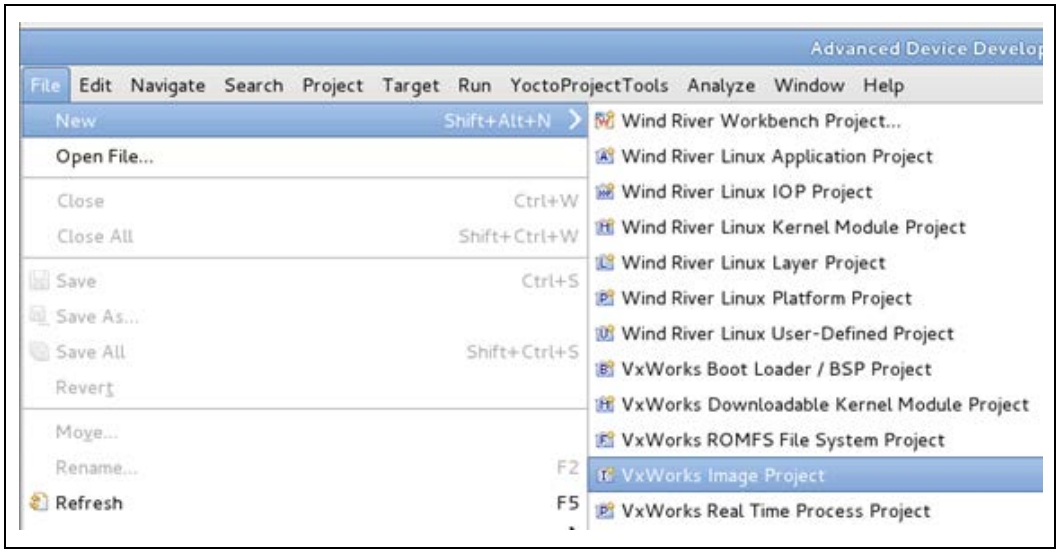
The procedure for creating VxWorks Guest OS #2 will be very similar to the steps for creating VxWorks #1.

NOTE: For more information see *Hypervisor User's Guide, section 16.3: Example: Hypervisor Standard Profile with a VxWorks Guest (Intel)*.

8.5.1 Create the second VxWorks Image Project (VIP)

Go to **File > New > VxWorks Image Project.** See Figure 82 – VxWorks Image Project.

Figure 82 – VxWorks Image Project

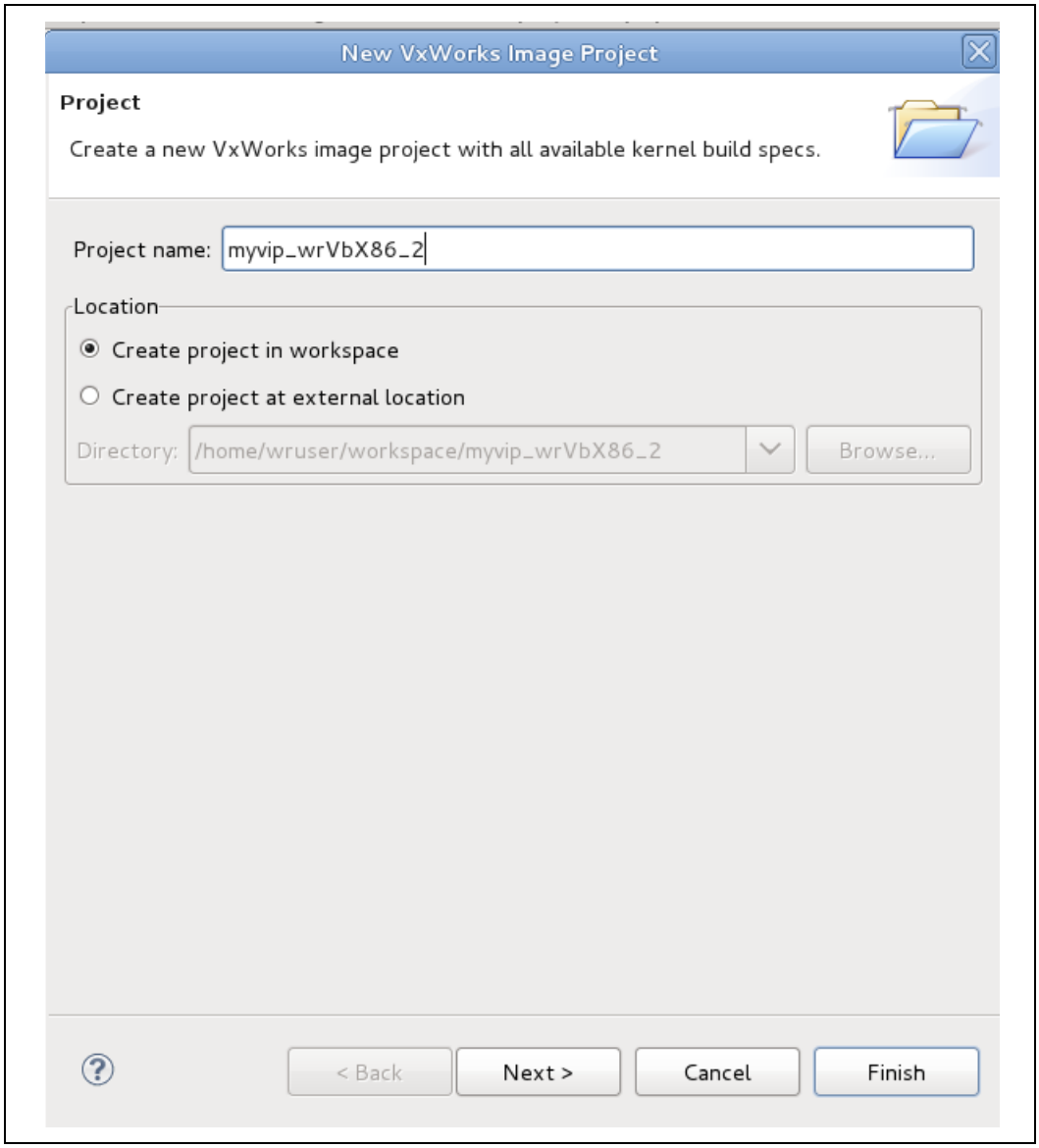


A **New VxWorks Image Project** window appears.

In the **Project name:** field, type **myvip_wrVbX86_2**

Select the **Create project in workspace** button. See Figure 83 – Create Project….

Figure 83 – Create Project…

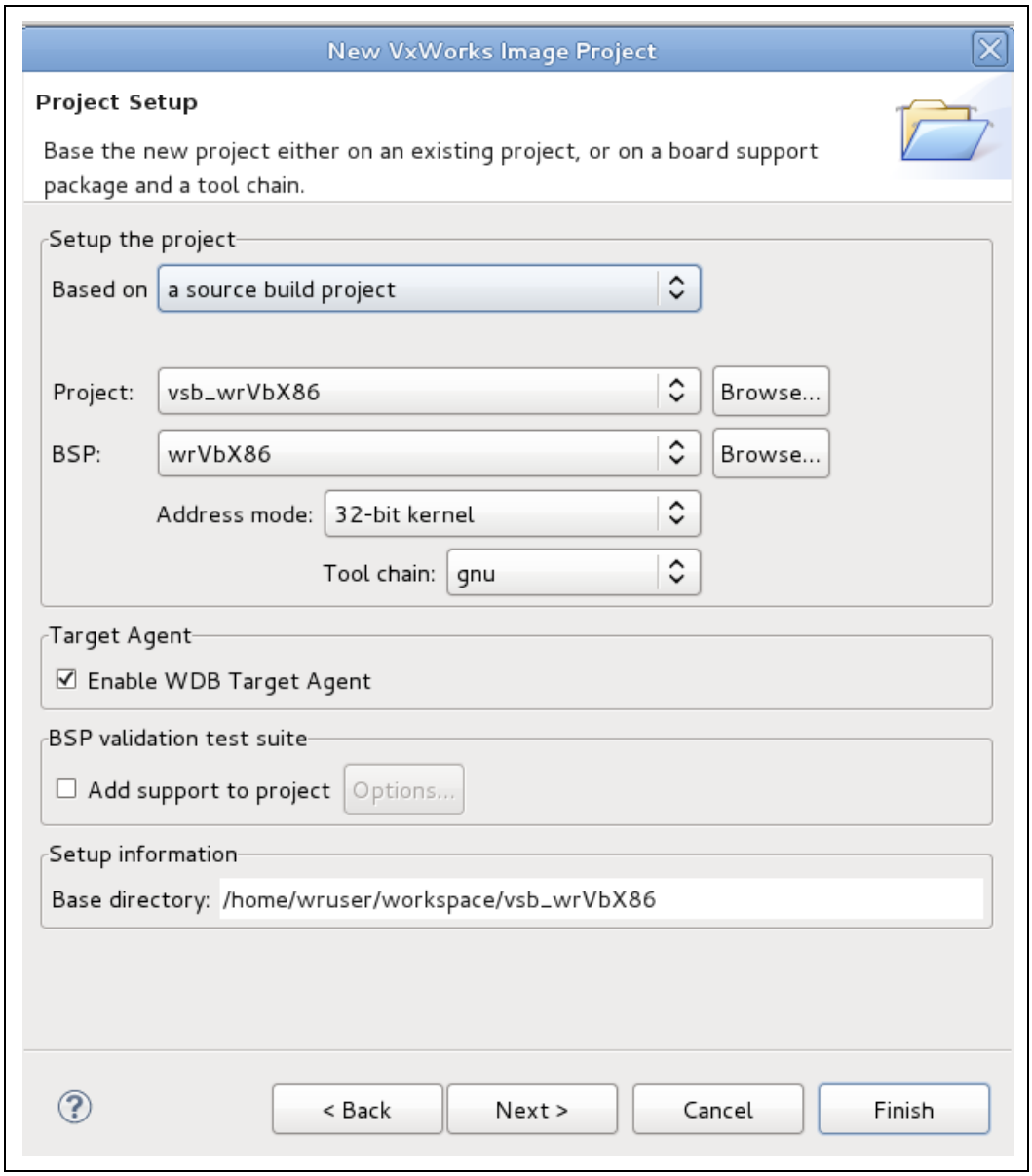


Click **Next**. The **Project Setup** window appears.

At the **Based on** drop-down menu, select **an existing project**.

From the **Project** drop-down menu select **myvip_wrVbX86_1**. See Figure 84 – Based On.

Figure 84 – Based On



Check the **Enable WDB Target Agent** checkbox.

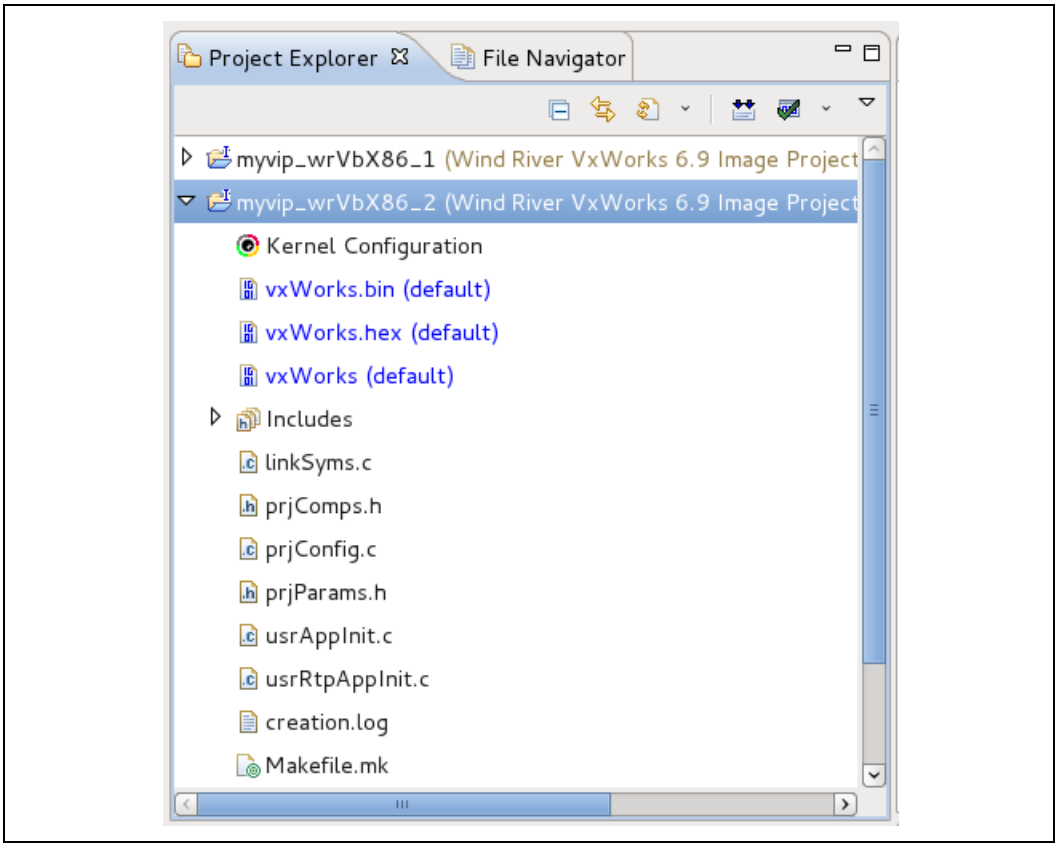
To create the VIP project, click **Finish**.

After a few moments, a **myvip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)** appears in the **Project Explorer** pane.

Go to Project Explorer.

Expand the **myvip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)**. Detailed component items appear below the image project line. See Figure 85 – Expanded Project.

Figure 85 – Expanded Project



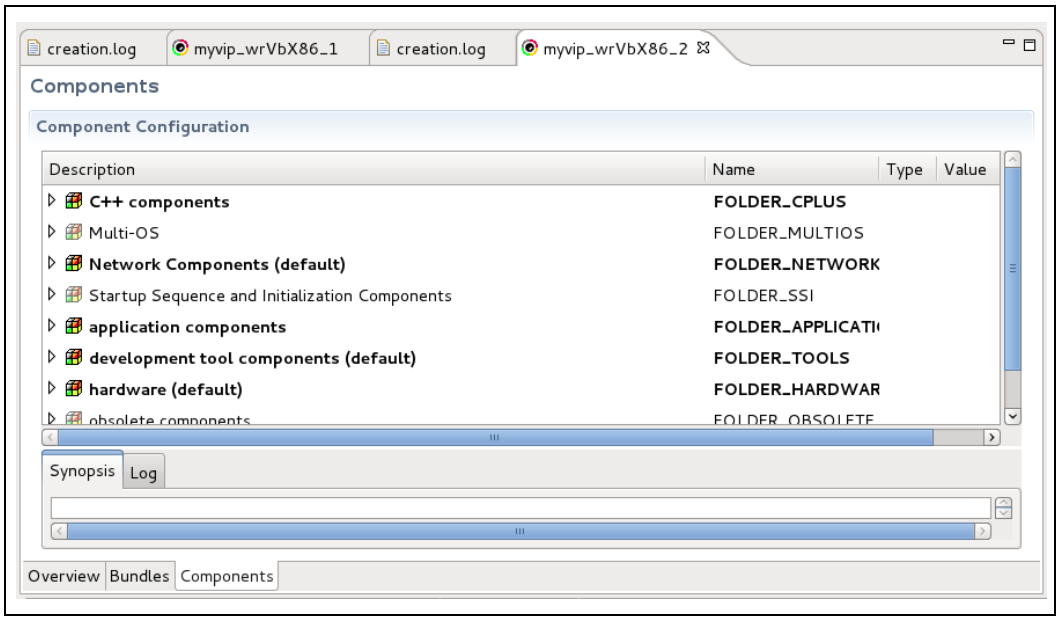
NOTE: If at this point, you do not wish to add components or applications to your VxWorks image project, proceed to Section 8.5.5 – Build the Project.

8.5.2 Add Components to VxWorks

Since **myvip_wrVbX86_2** is based on the **myvip_wrVbX86_1** project, most of the kernel components selected in section 2.1 will already be enabled. You simply need to add a few components to enable extra functionality.

Double-click the **Kernel Configuration** node. A **myvip_wrVbX86_2** Components tab appears in the editor pane. See Figure 86 - Components.

Figure 86 - Components



In the following steps, you may add the following list of components to the VIP. Add only those components you require.

IMPORTANT: The INCLUDE_RTL8169_VXB_END component listed below is a Realtek Ethernet driver for the VxWorks 2 OS. If you choose to install this driver, make certain to remove a conflicting driver by completing the steps that follow this notice, and then completing the procedure in Section 8.5.2.1 – Exclude DISK_UTIL.

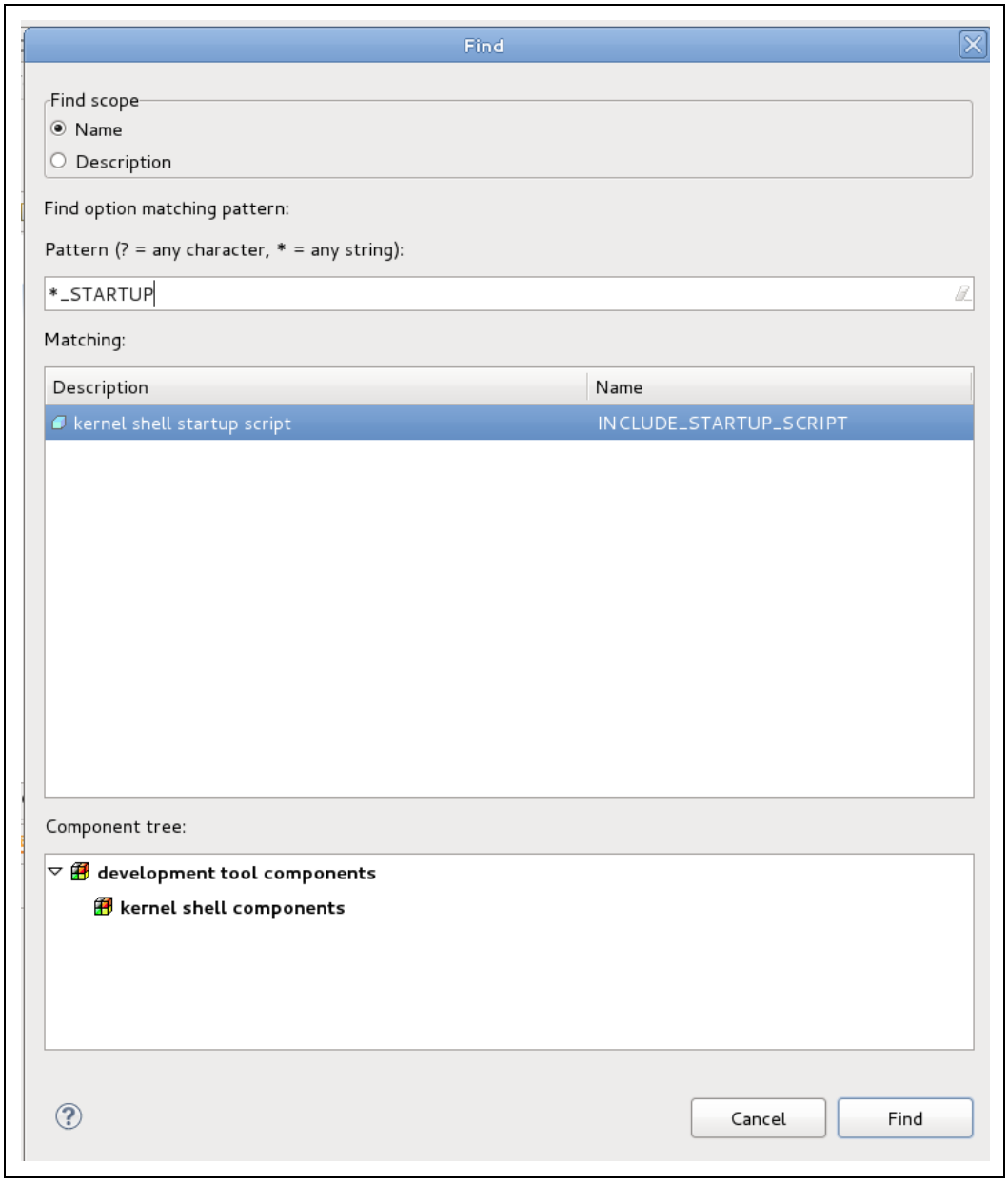
INCLUDE_STARTUP_SCRIPT
INCLUDE_ROMFS
INCLUDE_RTL8169_VXB_END

To add the list of components above, in the editor pane click the **myvip_wrVbX86_2** tab.

Press **Ctrl+F.** A **Find** window appears. This utility allows you to search for components in the Kernel Configuration Tool.

In the **Pattern** field, type the component from the list above that you wish to include in the kernel configuration. An object corresponding to the name you type appears in the **Matching:** pane. See Figure 87 - Name.

Figure 87 - Name



Double click the object in the **Matching:** window. The **Find** window closes and the newly added component appears as a highlighted item in the **Component Configuration** window.

In the **Component Configuration** window, right-click the component and select **Include (quick include)**. A progress window appears momentarily as the component

loads. The system returns to the **Component Configuration** window and the component you just added appears as a bolded item.

Repeat the procedure to add the remaining components to the VIP.

8.5.2.1 Exclude DISK_UTIL

If you opted to include the **INCLUDE_RTL8169_VXB_END** component from the previous procedure, you must exclude a conflicting driver (**INCLUDE_DISK_UTIL**) from the VxWorks project.

IMPORTANT: If you do not exclude this conflicting driver from the VxWorks project, then the VxWorks OS may not boot on the target platform.

Press **Ctrl+F.** A **Find** window appears. This utility allows you to search for components in the Kernel Configuration Tool.

In the **Pattern** field, type **INCLUDE_DISK_UTIL**. An object corresponding to the name you type appears in the **Matching:** pane.

Double click the object in the **Matching:** window. The **Find** window closes and the component appears as a highlighted item in the **Component Configuration** window.

In the **Component Configuration** window, right-click **INCLUDE_DISK_UTIL** and select **Exclude (quick exclude)**. A progress window appears momentarily as the component unloads. The system returns to the **Component Configuration** window and the component you just removed appears as an unbolded item.

Press **CTRL+S** to save the project.

Right click the **myvip_wrVbX86_2** object, and select **Rebuild Project**. The VxWorks project rebuilds.

8.5.3 Add User Applications to the VIP Using Real-Time Process (RTP)

You can use Real-Time Process (RTP) projects to manage, build, and incorporate applications in user space. Workbench provides pre-configured templates for various types of projects, and lets you create and import these project templates.

In this case, we will use an existing example RTP project to create a simple Hello World executable.

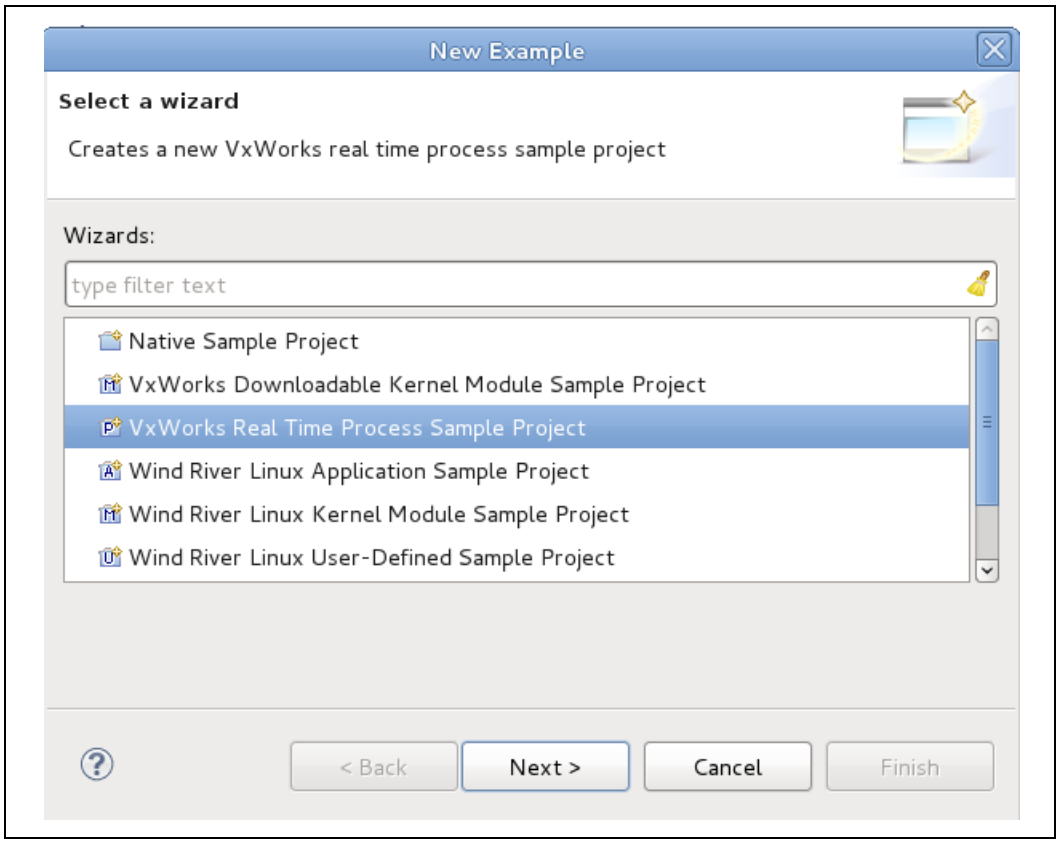
Once your development work is complete, RTPs can be incorporated into a ROMFS project to create a file system embedded in the VxWorks project. The following sections provide instruction to add your application into the kernel.

To create a RTP project based on an the hello world example, perform the following procedure:

Select **File > New > Example...**

Highlight VxWorks Real Time Process Sample Project and click **Next**. See Figure 88 – VxWorks RTP.

Figure 88 – VxWorks RTP

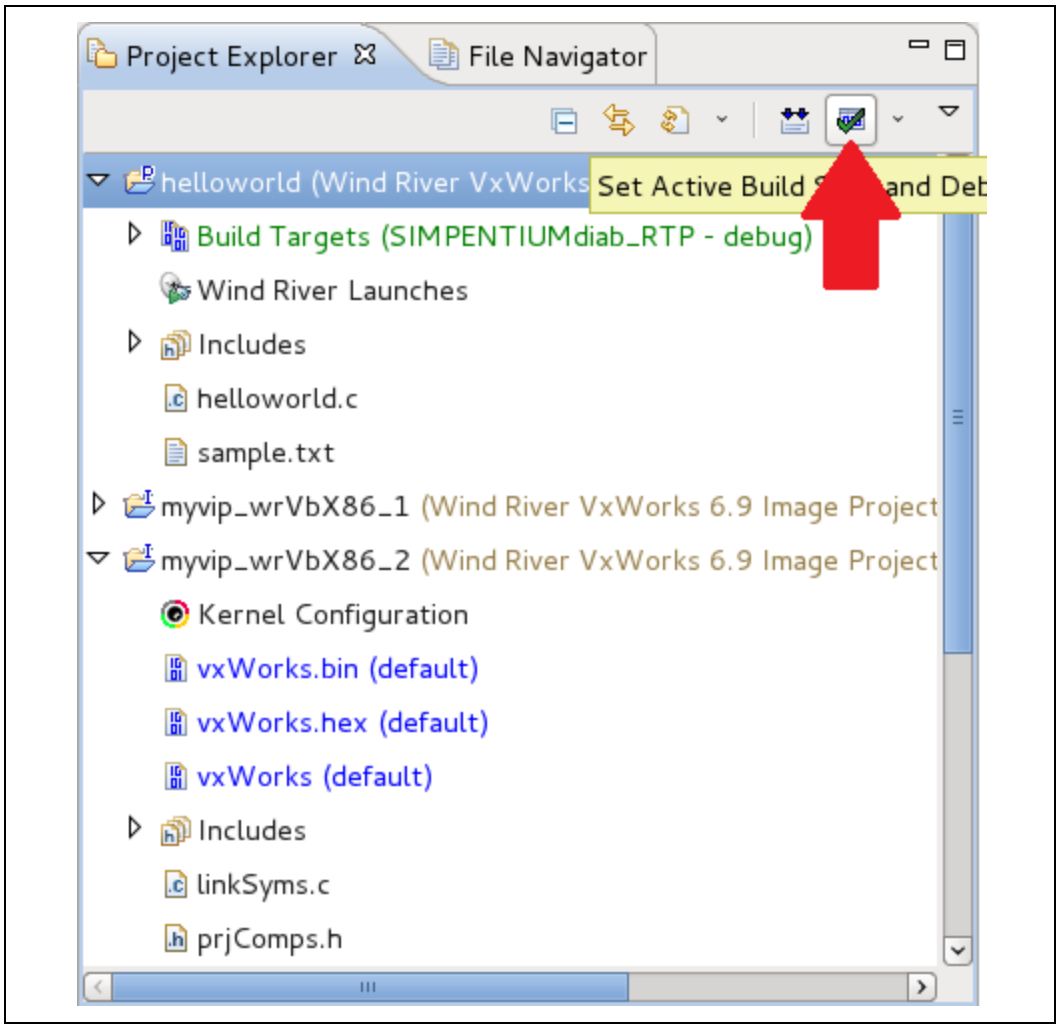

New Example
Select a wizard
Creates a new VxWorks real time process sample project
Wizards:
type filter text
Native Sample Project
VxWorks Downloadable Kernel Module Sample Project
VxWorks Real Time Process Sample Project
Wind River Linux Application Sample Project
Wind River Linux Kernel Module Sample Project
Wind River Linux User-Defined Sample Project
< Back
Next >
Cancel
Finish

Select **The Hello World Demonstration Program**.

Click **Finish**. A new project called **helloworld** appears in the Workbench Project Explorer window.

Select the project and click the **Set Active Build Spec and Debug Mode** icon. See the red-arrowed item in Figure 89 – Set Active….

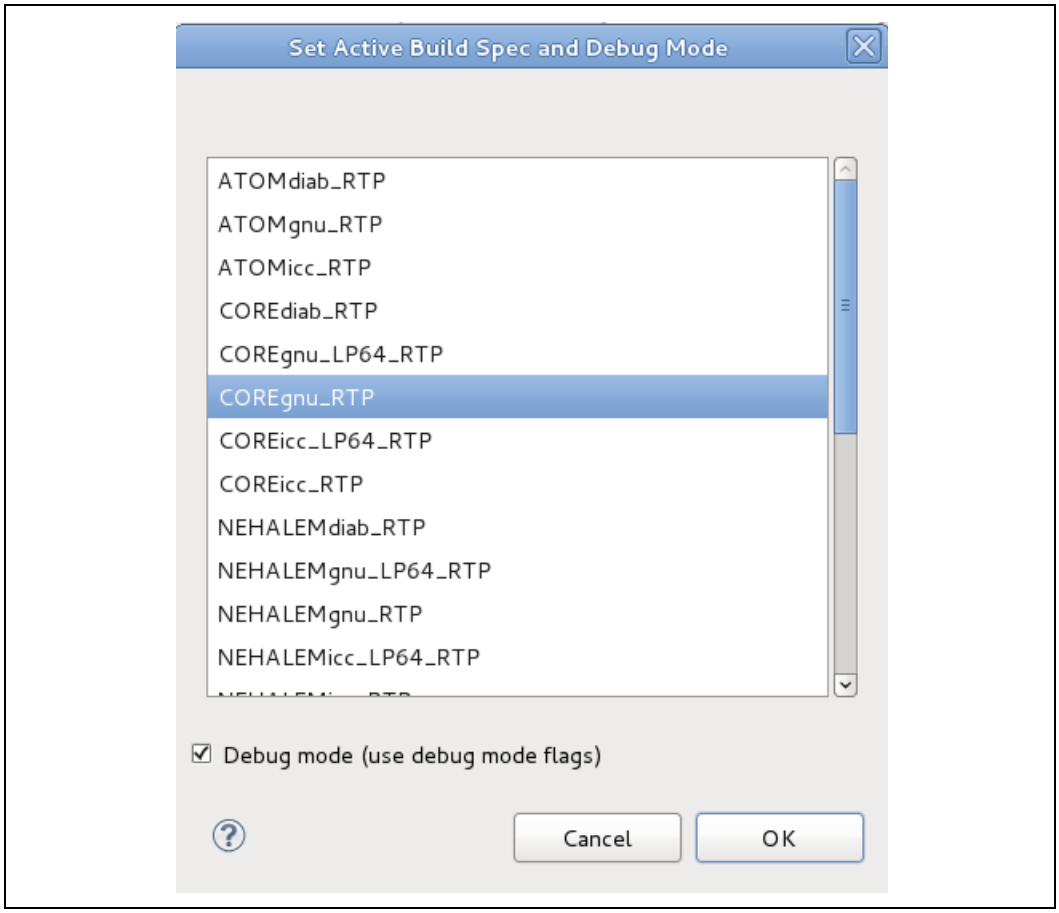
Figure 89 – Set Active…



The **Set Active Build Spec and Debug Mode** window appears.

Select the **COREgnu_RTP** build spec. See Figure 90 - COREgnu_RTP.

Figure 90 - COREgnu_RTP



Click **OK**. A popup window appears asking **Do you wish to rebuild the index now?**

Click **Yes**.

Right-click the **helloworld** project and select **Build Project**.

8.5.4 Create a ROMFS File System Project

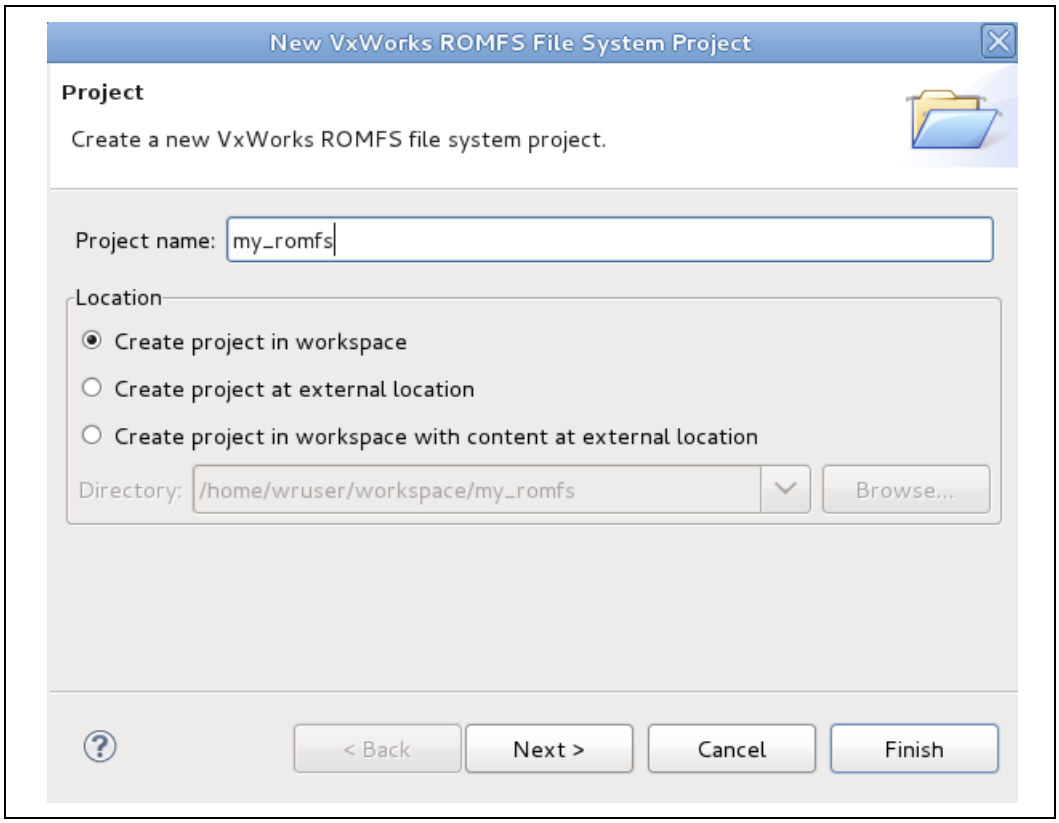
A Real-Time Project (RTP) application is similar to a regular executable program. It must exist on a file system from where it can be executed. A read-only memory file system (ROMFS) project acts as a read-only file system that can be embedded in a VxWorks image.

We will include our RTP executable in the ROMFS project which will itself be included in the VxWorks Image Project.

To create a ROMFS project, perform the following procedure:

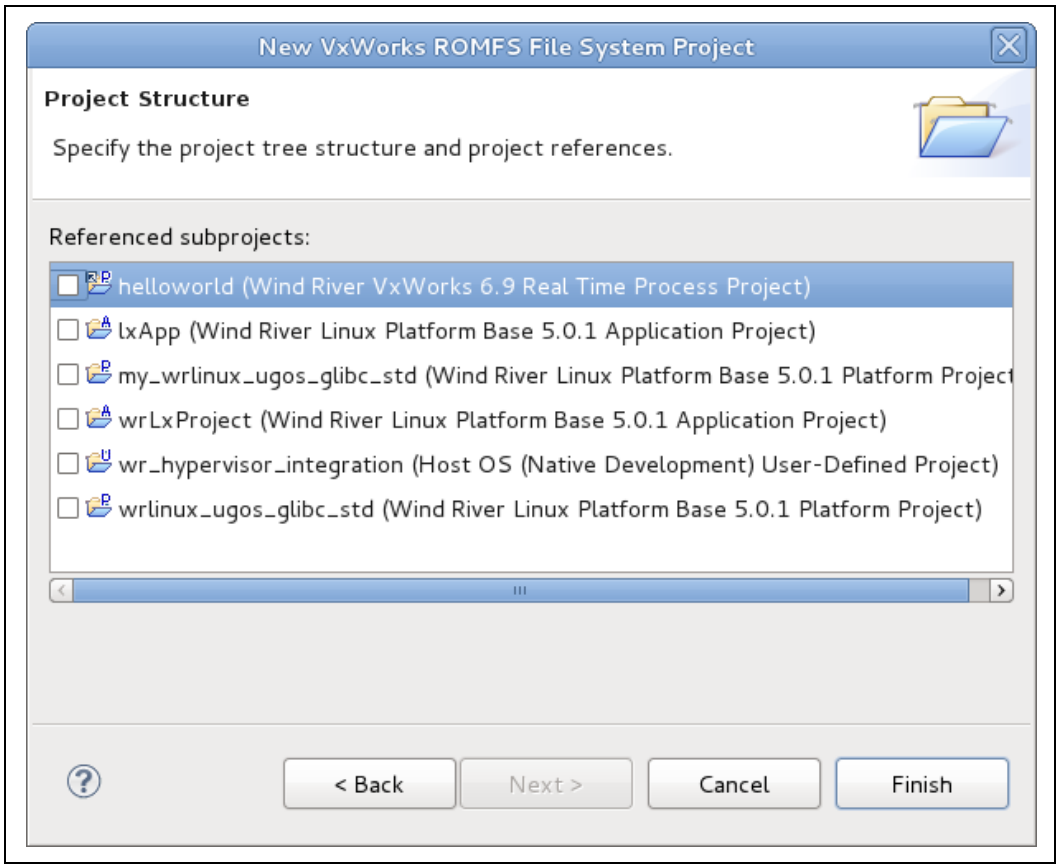
Select **File > New > VxWorks ROMFS File System Project**. A New VxWorks ROMFS File System Project window appears. See Figure 91 - my_romfs.

Figure 91 - my_romfs



Set the project name to **my_romfs** and select **Next**. A **New VxWorks ROMFS...** window appears. See Figure 92 - New VxWorks ROMFS.

Figure 92 - New VxWorks ROMFS

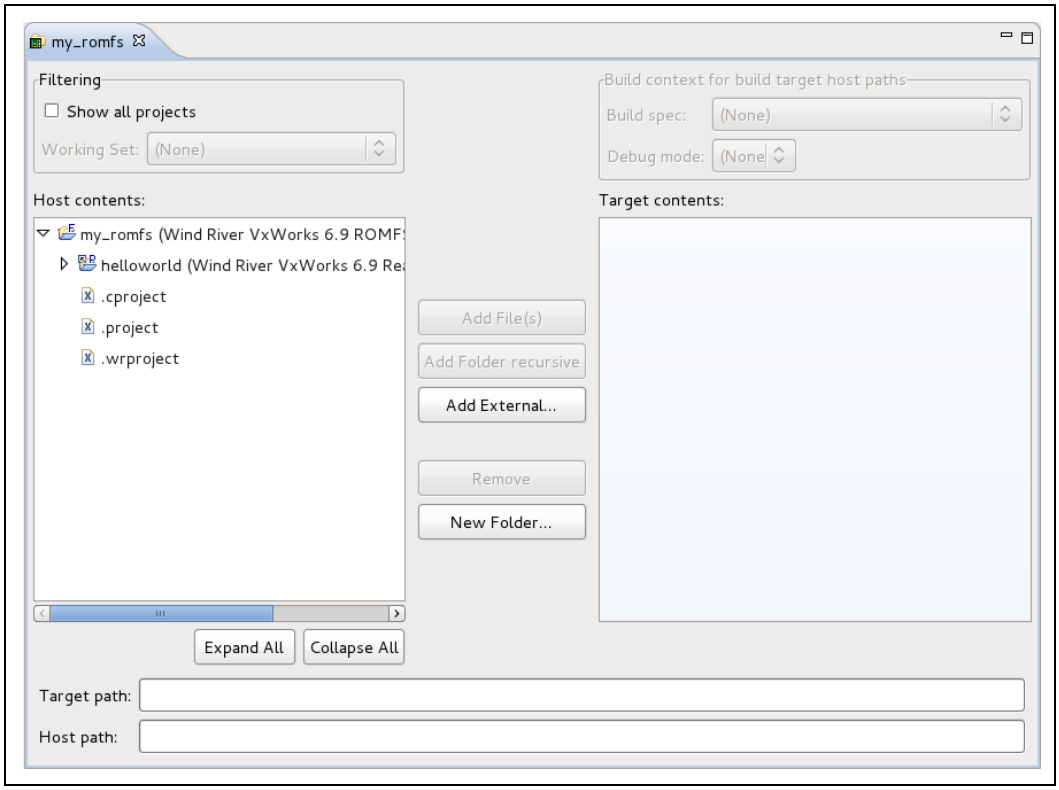

New VxWorks ROMFS File System Project
Project Structure
Specify the project tree structure and project references.
Referenced subprojects:
helloworld (Wind River VxWorks 6.9 Real Time Process Project)
lxApp (Wind River Linux Platform Base 5.0.1 Application Project)
my_wrlinux_ugos_glibc_std (Wind River Linux Platform Base 5.0.1 Platform Project
wrLxProject (Wind River Linux Platform Base 5.0.1 Application Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)
wrlinux_ugos_glibc_std (Wind River Linux Platform Base 5.0.1 Platform Project)
< Back
Next >
Cancel
Finish

In the Project Structure wizard's **Referenced subprojects** panel, select the **helloworld** project.

Click **Finish**.

A new editor panel opens with a tab titled **my_romfs**. See Figure 93 – my_romfs Tab.

Figure 93 – my_romfs Tab



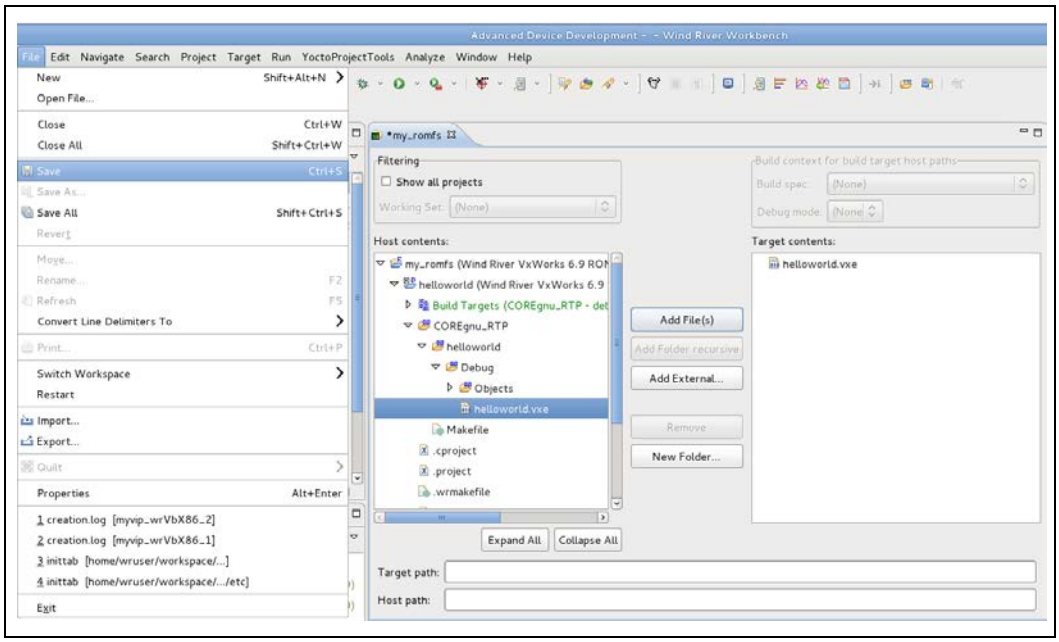
In the host contents pane, open the **helloworld** project node by clicking on the triangle.

Descend into the file tree by clicking the triangle icons in front of the directory names: **my_romfs > helloworld > COREgnu_RTP > helloworld > Debug > Objects**.

Select the **helloworld (helloworld.vxe)** item and click **Add File(s)**.

From the Workbench menu select **File > Save** to save the changes to the ROMFS. See Figure 94 - helloworld.exe.

Figure 94 - helloworld.exe



The ROMFS will now contain a copy of the executable. Whenever the executable is recompiled, the ROMFS will be regenerated with the updated file.

8.5.5 Build the Project

After you have created and configured your project, you must build it. In this case, we will embed the ROMFS within the VxWorks kernel itself.

NOTE: For more information on projects and project structure, see *Wind River Workbench User's Guide: Projects Overview*, a separate document you may access through Workbench Help.

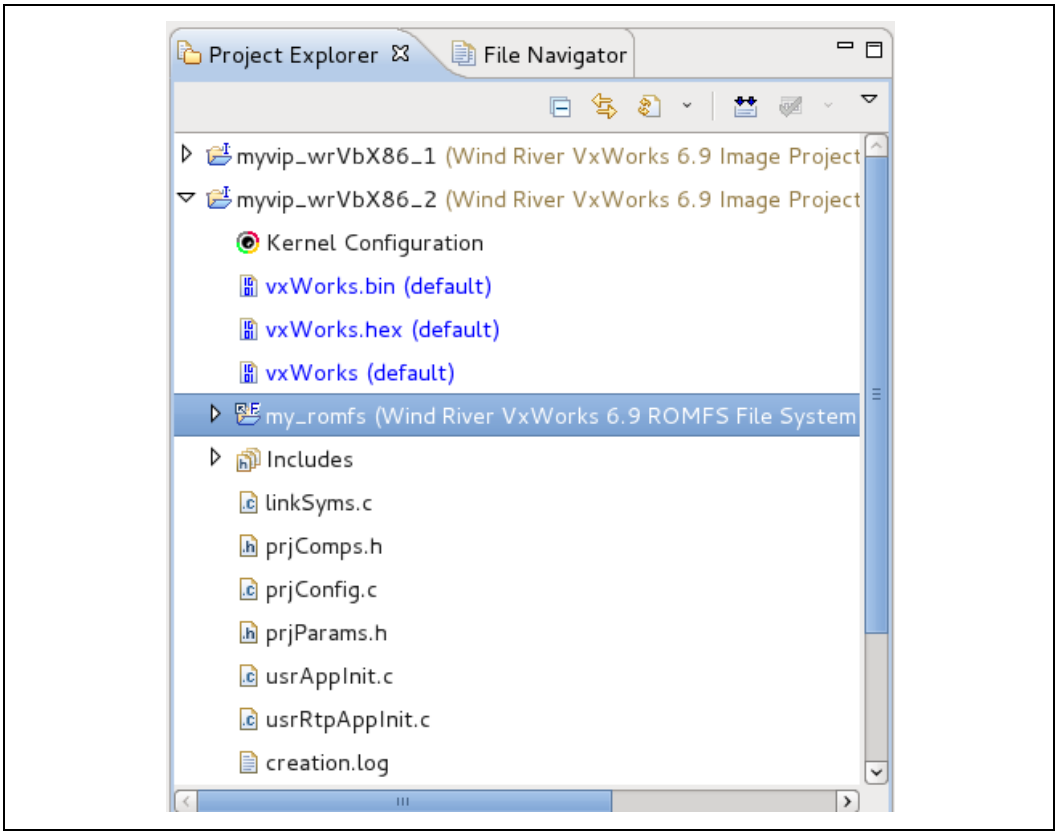
You can build VxWorks kernel applications and associated projects by adding your ROMFS project to the VIP and then build it.

To add a ROMFS project to the VIP, follow this procedure:

Right-click the ROMFS project **my_romfs** in the **Project Explorer** view, select **Project References > Add as Project Reference**. An **Add as a Project Reference** window appears.

Select **myvip_wrVbX86_2** and click **OK**. In File Explorer, the ROMFS project object is now embedded under the **myvip_wrVbX86_2** project. See Figure 95 - ROMFS Project.

Figure 95 - ROMFS Project

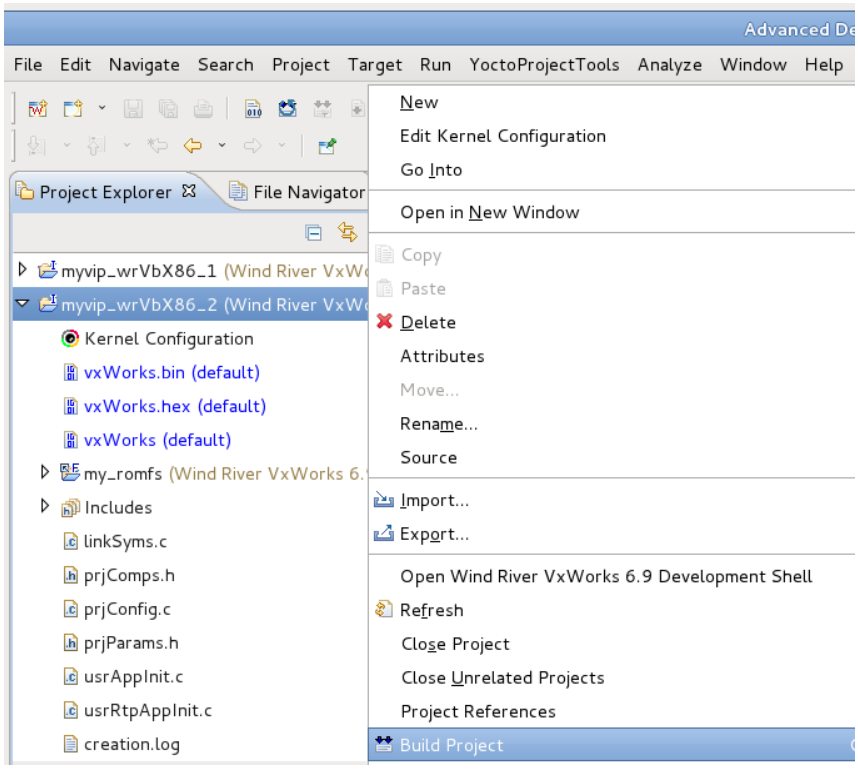


8.5.6 Build the VIP

To build the VxWorks Image Project (VIP), go to the **Project Explorer** pane and right-click **myvip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)**. A drop-down menu appears.

Select **Build Project**. See Figure 96 – Build Project.

Figure 96 – Build Project



The build window shows lines of code compiling. A build project progress window appears, and a progress bar moves repeatedly left and right while the system compiles the VxWorks libraries required for the Guest OS.

When the project finishes building, a **Build Finished in Project vip_wrVbX86_2** statement appears at the bottom of the **Build Console** pane.

8.6 Create and Build WR Linux

Follow the procedure below to create and build a Wind River Linux project.

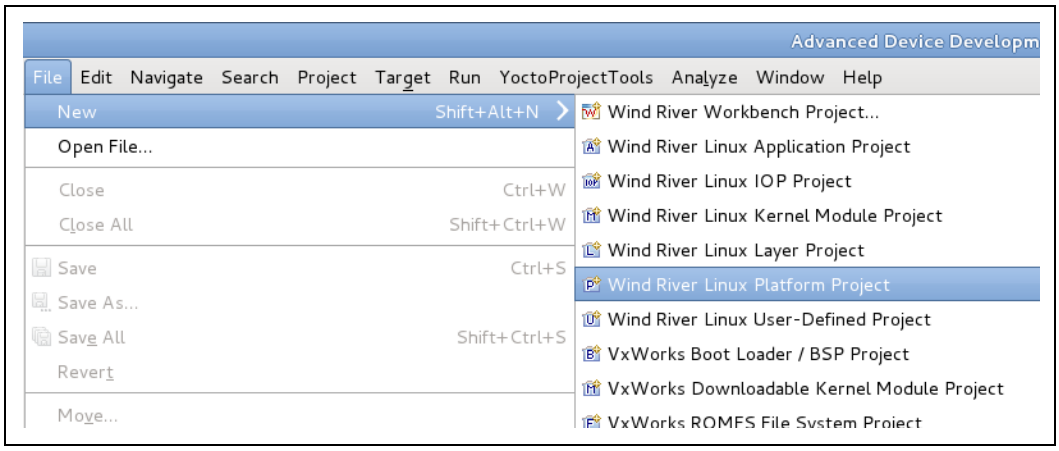
8.6.1 Create a WR Linux Platform Project

On the development host open Wind River Workbench.

Switch to the Advanced Device Development perspective: **Window > Open Perspective > Advanced Device Development**.

Go to **File > New > Wind River Linux Platform Project.** See Figure 97 – Linux Project.

Figure 97 – Linux Project

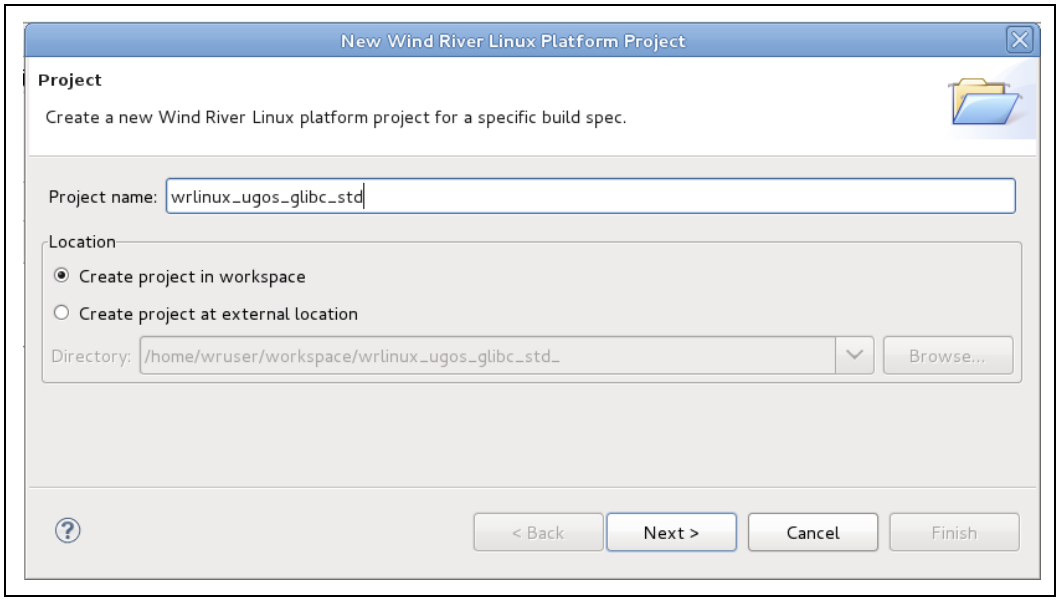

Advanced Device Developm
File Edit Navigate Search Project Target Run YoctoProjectTools Analyze Window Help
New Shift+Alt+N
Open File...
Close Ctrl+W
Close All Shift+Ctrl+W
Save Ctrl+S
Save As...
Save All Shift+Ctrl+S
Revert
Move...
Wind River Workbench Project...
Wind River Linux Application Project
Wind River Linux IOP Project
Wind River Linux Kernel Module Project
Wind River Linux Layer Project
Wind River Linux Platform Project
Wind River Linux User-Defined Project
VxWorks Boot Loader / BSP Project
VxWorks Downloadable Kernel Module Project
VxWorks ROMFS File System Project

After a few moments a **New Wind River Platform Project** window appears.

In the **Project name:** field, type **wrlinux_ugos_glibc_std** (or whatever name you choose to call the project). This procedure uses the project name **wrlinux_ugos_glibc_std**.

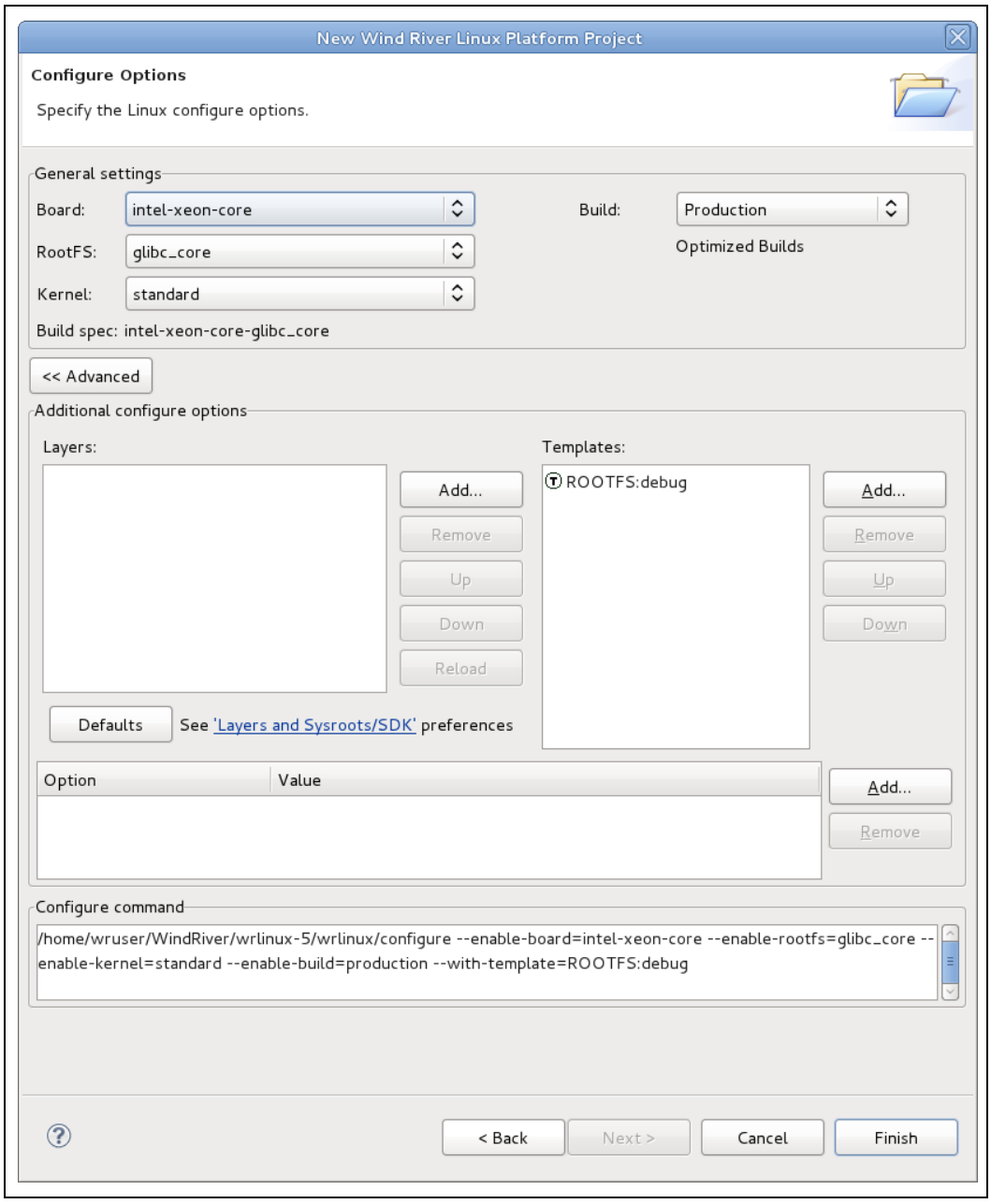
Select the **Create project in workspace** button. See Figure 98 - Project.

Figure 98 - Project



Click **Next**. A **Configure Options** window appears. See Figure 99 – Configure Options 1.

Figure 99 – Configure Options 1



Under **General settings**, select values as follows:

- **Board –** From the drop-down menu, select **intel-xeon-core-32**.
- **RootFS** (Root File System) **–** From the drop-down menu, select **glibc_std** to enable a small EGLIB based file system that does not use busybox.
- **Kernel –** From the drop-down menu, select **standard** to enable a standard Wind River Linux kernel.

From the **Build** drop-down menu, select **Production**.

The **Options | Value** pane will retain the list options you configured for the previous Wind River Linux Platform Project. If this is your first platform project it will be empty. See Figure 99 – Configure Options 1.

NOTE: In some low-resolution monitor hardware, you may need to move the scroll bar or reposition the window to view all of the window's contents.

At the bottom of the **Configure Options** window, there is an **Option | Value** pane.

At the right of the **Option | Value** pane use the **Add** and **Remove** buttons to configure the list with the following options:

--enable-parallel-pkgbuilds=4

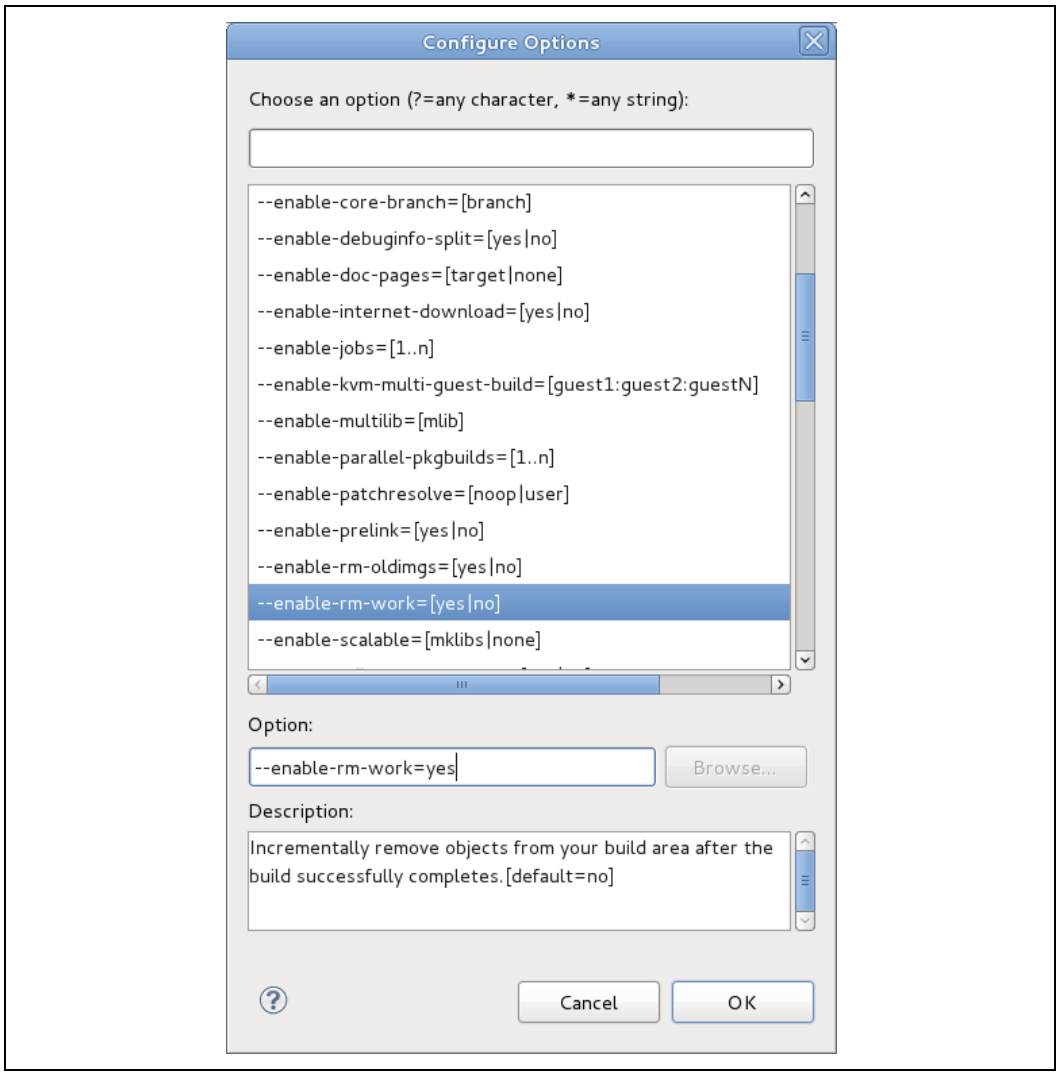
--enable-rm-oldimgs=yes

--enable-rm-work=yes

--with-package=inetutils-telnetd,git,openssl,openssh,grub

Click the **Add** button. A **Configure Options** window appears. See Figure 100 – Configure Options 2.

Figure 100 – Configure Options 2



Scroll the list and select the option, for example; **--enable-parallel-pkgbuilds=[1..n]**

NOTE: Most of these options optimize the speed and size of the platform build process and do not affect the contents of the kernel or file system built. They may be modified as necessary for your build host. Use the following steps to determine the number of processor cores your development host hardware has.

NOTE: In the **Option** text box, replace any brackets and their contents with the desired value. For example; replace **[yes|no]** with **yes.**

In Workbench go to **Applications > System Tools > System Monitor**. A **System Monitor** window appears.

Count the number of processor cores your development host has.

Enter the number of processor cores into the processor core value space in the following string:

--enable-parallel-pkgbuilds=[NNNNNNN]

For example, if your development host has a 4-core processor, modify the entry to:

--enable-parallel-pkgbuilds=4

if your development host has a 16-core processor, modify the entry to:

--enable-parallel-pkgbuilds=16

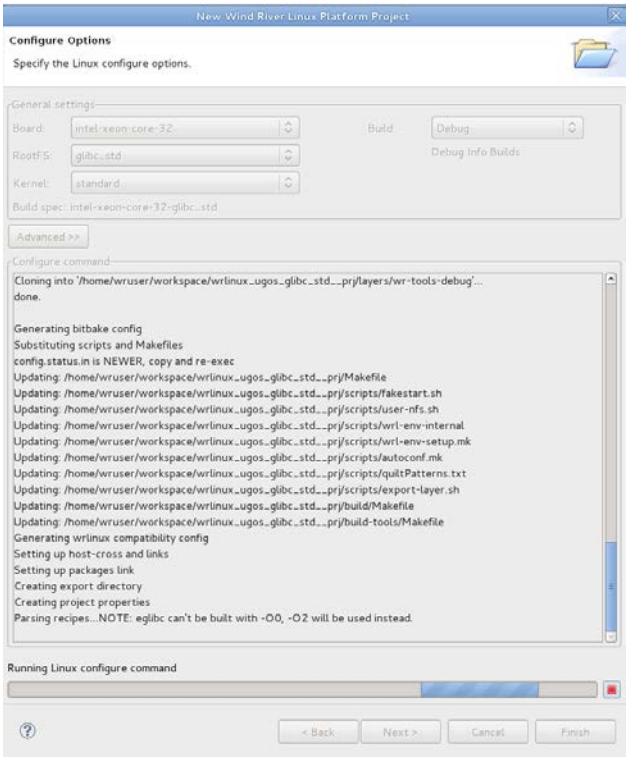
Click **OK**.

The **Configure Options** window closes.

Click **OK** and click **Finish**. Inside the **Configure Options** window, code lines scroll through the **Configure command** pane, and a progress bar moves repeatedly left and right as the system creates the project. See Figure 101 - Processing.

NOTE: Processing may take several minutes.

Figure 101 - Processing



When the system finishes, the **Configure Options** window closes. In the **Project Explorer** window a **wrlinux_ugos_glibc_std (Wind River Linux Platform Base 5.0.1 Platform Project)** object appears.

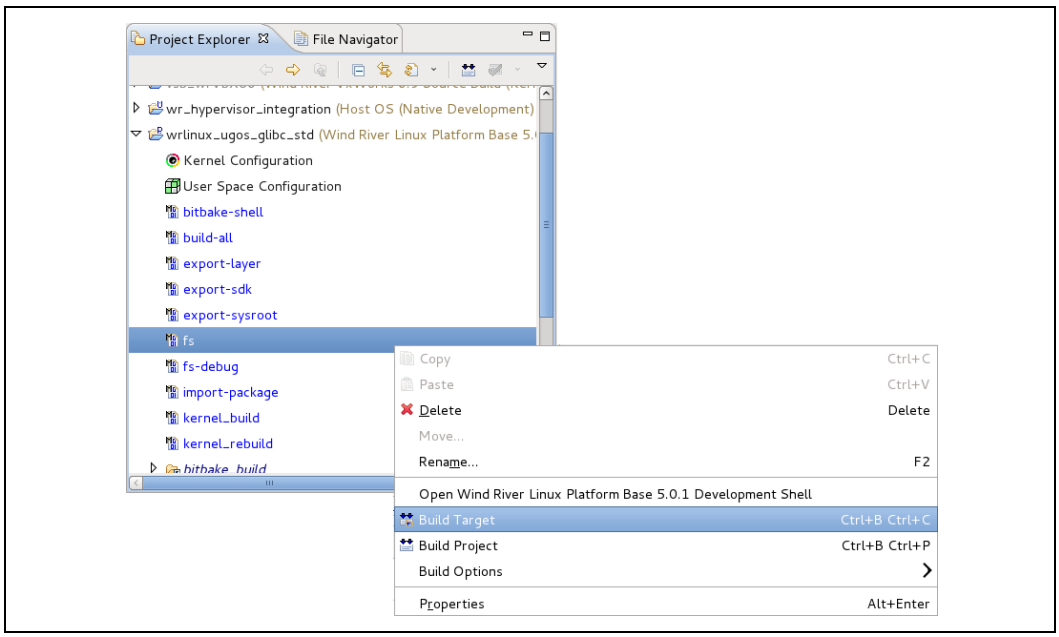
8.6.2 Configure and Build the File System and Packages

In this section you configure the target platform file system. To do this you must create an initial build of the kernel.

In **Project Explorer** , double-click **wrlinux_ugos_glibc_std (Wind River Linux Platform Base 5.0.1 Platform Project)** object. The project expands.

Right-click the **fs** object. An options list appears. See Figure 102 – Build Target.

Figure 102 – Build Target



Select **Build Target**. A **Build Target – kernel_build – project 'wrlinux_ugos_glibc_std '** progress window appears. At the bottom of the Workbench, lines of code scroll through the **Build Console** tab.

NOTE: Depending on your system's speed, the build process may require many minutes. Build time on some systems may last several hours.

Open a terminal window from your desktop. Optionally click inside the **Workbench Project Explorer** view and select Open **Wind River Platform Base 5.0.1 Development Shell** from the menu.

At the terminal prompt type:

```
$ cd /home/wruser/workspace/wrlinux_ugos_glibc_std_prj/

$ make ARCH=x86
```

Close the terminal window.

```
$ exit
```

In **Project Explorer** under the **wrlinux_ugos_glib_std** project, double-click the **User Space Configuration** node to open it in the editor.

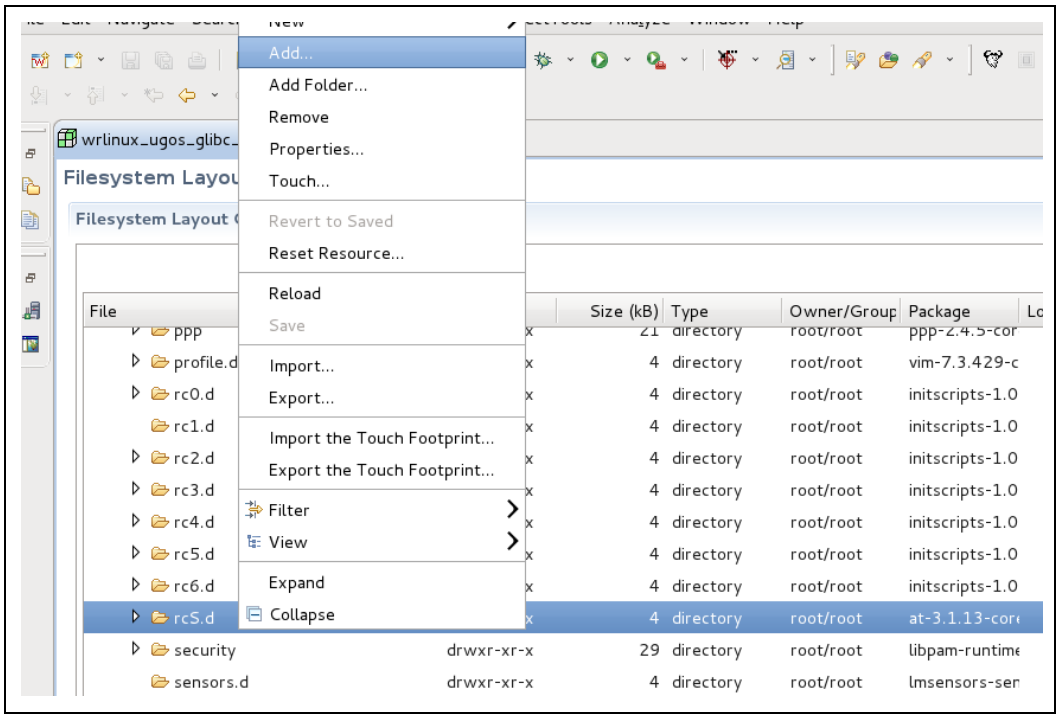
NOTE: If a dialog states **Would you like to rebuild the package database for Workbench**, then click **OK** to continue. Otherwise proceed to the next step.

Double-click the **wrlinux_ugos_glib_std** project. The **Filesystem Layout** editor appears.

Expand the file tree, and right click the **/etc/rcS.d** directory.

Select **Add** from the menu. See Figure 103 - Add.

Figure 103 - Add



In the file browser dialog, navigate to:

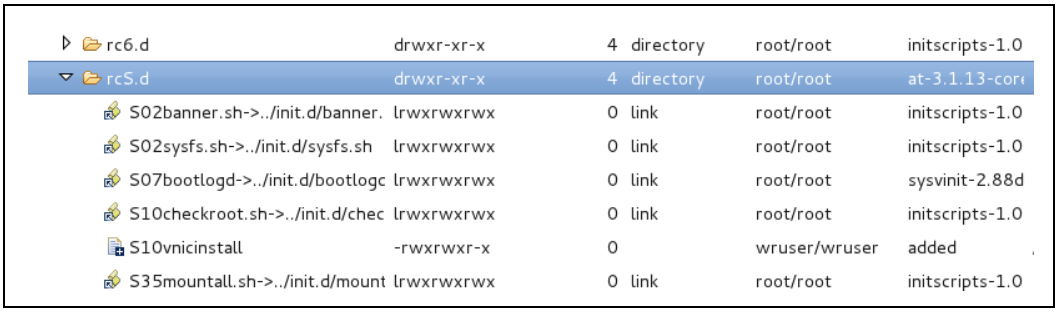
/home/wruser/workspace/vnic_module/

and select **S10vnicinstall**

Click **OK**.

The file appears as shown in Figure 104 – File Content.

Figure 104 – File Content


rc6.d drwxr-xr-x 4 directory root/root initscripts-1.0
rcS.d drwxr-xr-x 4 directory root/root at-3.1.13-cor
S02banner.sh->../init.d/banner. lrwxrwxrwx 0 link root/root initscripts-1.0
S02sysfs.sh->../init.d/sysfs.sh lrwxrwxrwx 0 link root/root initscripts-1.0
S07bootlogd->../init.d/bootlogc lrwxrwxrwx 0 link root/root sysvinit-2.88d
S10checkroot.sh->../init.d/chec lrwxrwxrwx 0 link root/root initscripts-1.0
S10vnicinstall -rwxrwxr-x 0 wruser/wruser added
S35mountall.sh->../init.d/mount lrwxrwxrwx 0 link root/root initscripts-1.0

In the **Filesystem Layout** configuration window right-click the **/etc/inittab** object.

Select **Remove**.

In the **Filesystem Layout** configuration window right-click the **/etc** directory.

Select **Add** from the menu. A window appears.

In the file browser dialog, go to:

/home/wruser/workspace/custom_files/

Select the **inittab** object and click **OK**.

In the **Filesystem Layout** editor right click on **/lib** directory. Select **Add** from the menu.

In the file browser dialog, go to:

/home/wruser/workspace/vnic_module/vbi/

Select the **vbi.ko** object and click **OK**.

Right-click the **/lib** directory again. Select **Add** from the menu.

In the file browser dialog, go to:

/home/wruser/workspace/vnic_module/vnic/

Select the **vnic.ko** object and click **OK**.

From the Workbench menu select **File > Save**.

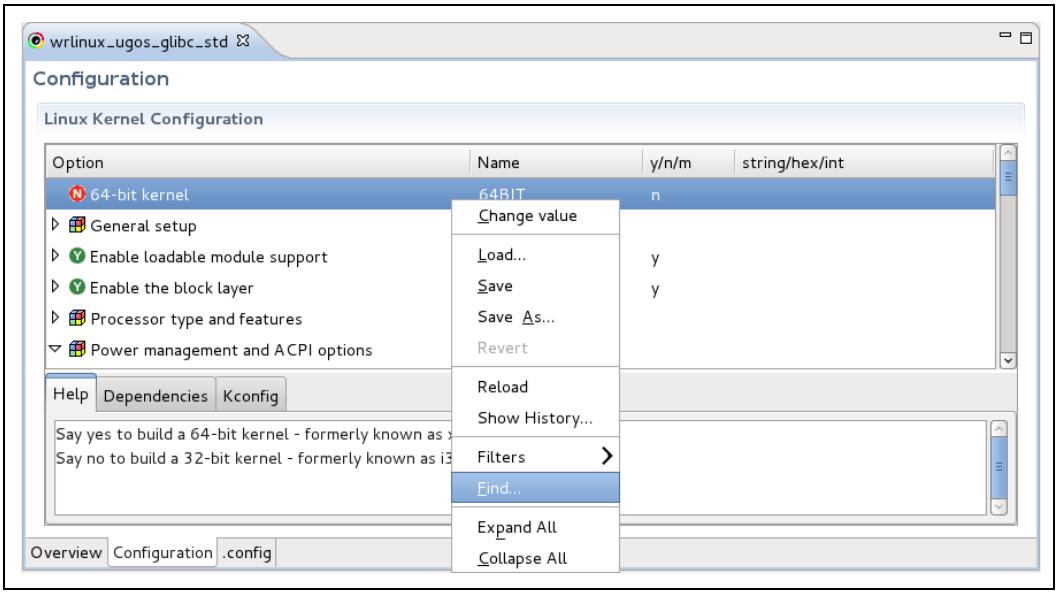
In **Project Explorer**, double-click **Kernel Configuration.**

NOTE: If a dialog appears click **YES** to continue.

The kernel sources unpack. This may take several minutes. The **Linux Kernel Configuration** editor appears.

Right-click in the editor and select **Find**. See Figure 105 - Find.

Figure 105 - Find



In the **Pattern** text box, check the **Show All** check box.

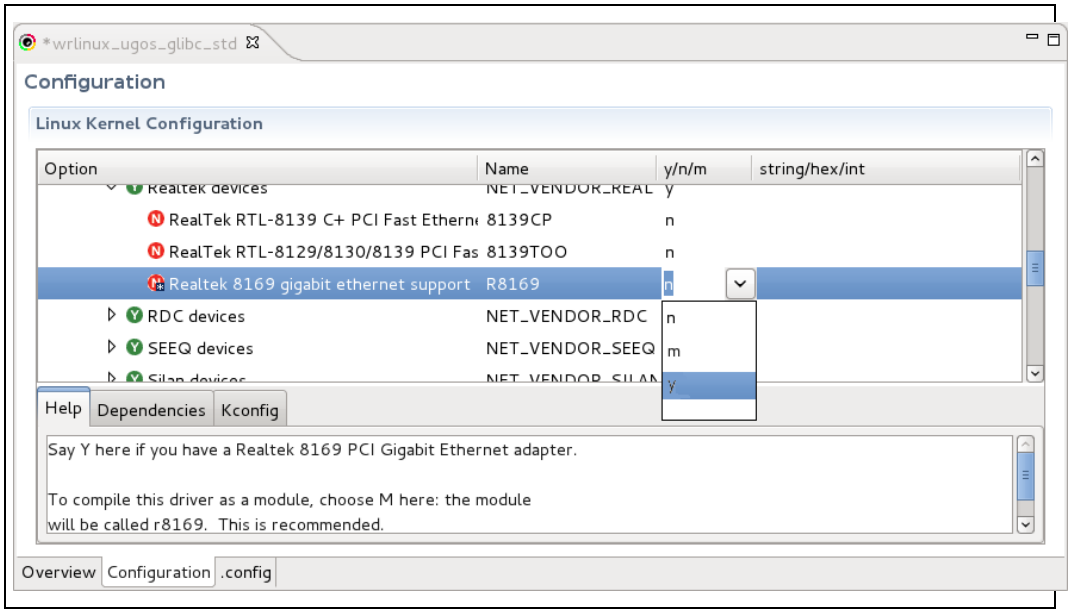
In the **Find** window enter **r8169**.

Double-click on **Realtek 8169 gigabit ethernet support** in the **Matching**: list box.

The **Find** window closes and the **R8169** configuration line is highlighted within the editor.

In the **y / n / m** column, select **y** from the dropdown menu. See Figure 106 – y/n/m.

Figure 106 – y / n / m



From the Workbench menu select **File > Save**.

In **Project Explorer** under the **wrlinux_ugos_glib_std** project, right-click **fs** and select **Build target** to make the file system and kernel with the changes.

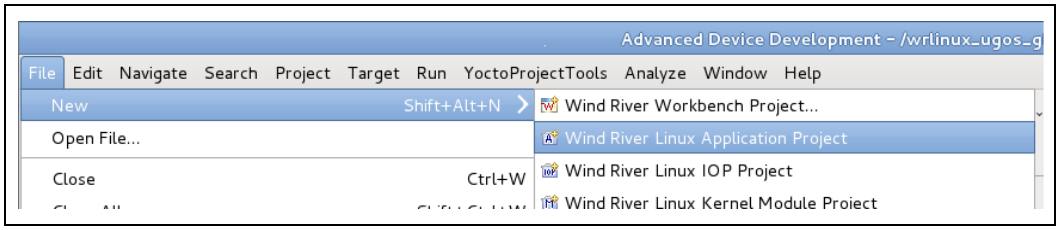
NOTE: Depending on your system's speed, the process may require many minutes.

8.6.3 Add a User Application to Linux

Go to **File → New → Wind River Linux Application Project**. See Figure 107 - Wind River Application Project.

NOTE: This menu item is only available in the Advanced Device Development perspective you selected at the beginning of Section 8.6.1 – Create a WR Linux Platform Project.

Figure 107 - Wind River Application Project



A **New Wind River Linux Application Project** window appears.

In the **Project Name:** field type **lxApp**

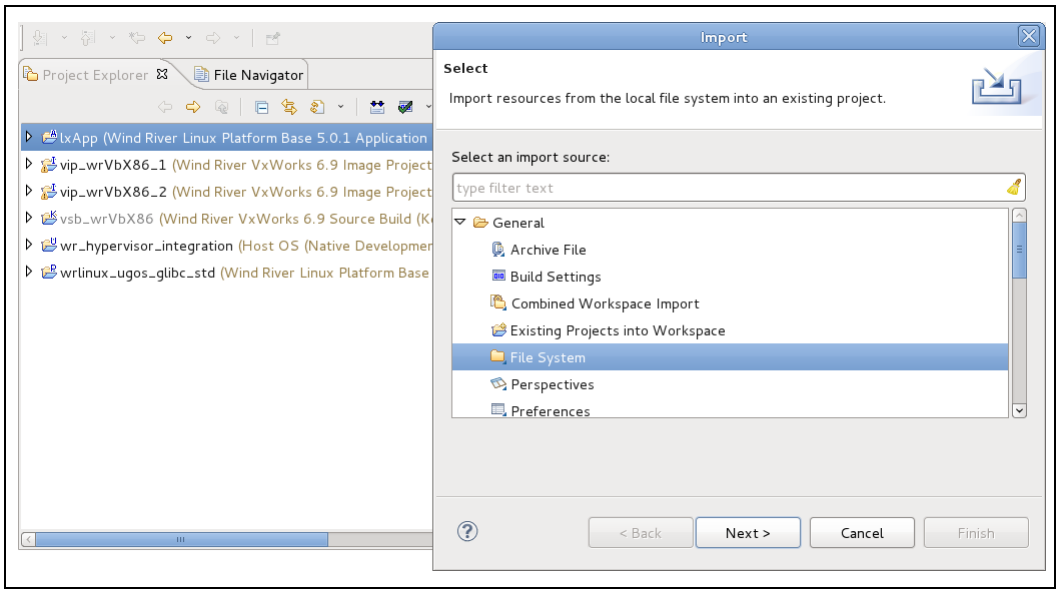
NOTE: The first letter in the **lxApp** project name is lower-case “L”.

Click **Finish**. An **lxApp (Wind River Linux Platform Application Project)** object appears in **Project Explorer**.

In **Project Explorer**, right-click **lxApp (Wind River Linux Platform Application Project)** and select **Import**. A **Select** window appears.

In the **Select an import source:** list, expand the **General** icon. The icon options expand beneath. See Figure 108 – General Icon.

Figure 108 – General Icon



Select **File System** and click **Next**. An **Import** window opens.

Adjacent to the **From directory:** field, click **Browse…** and browse to your application directory.

NOTE: In the following step this workflow example imports an application called “hello.c”. To import your own code, substitute the directory path below with the directory to your own source code.

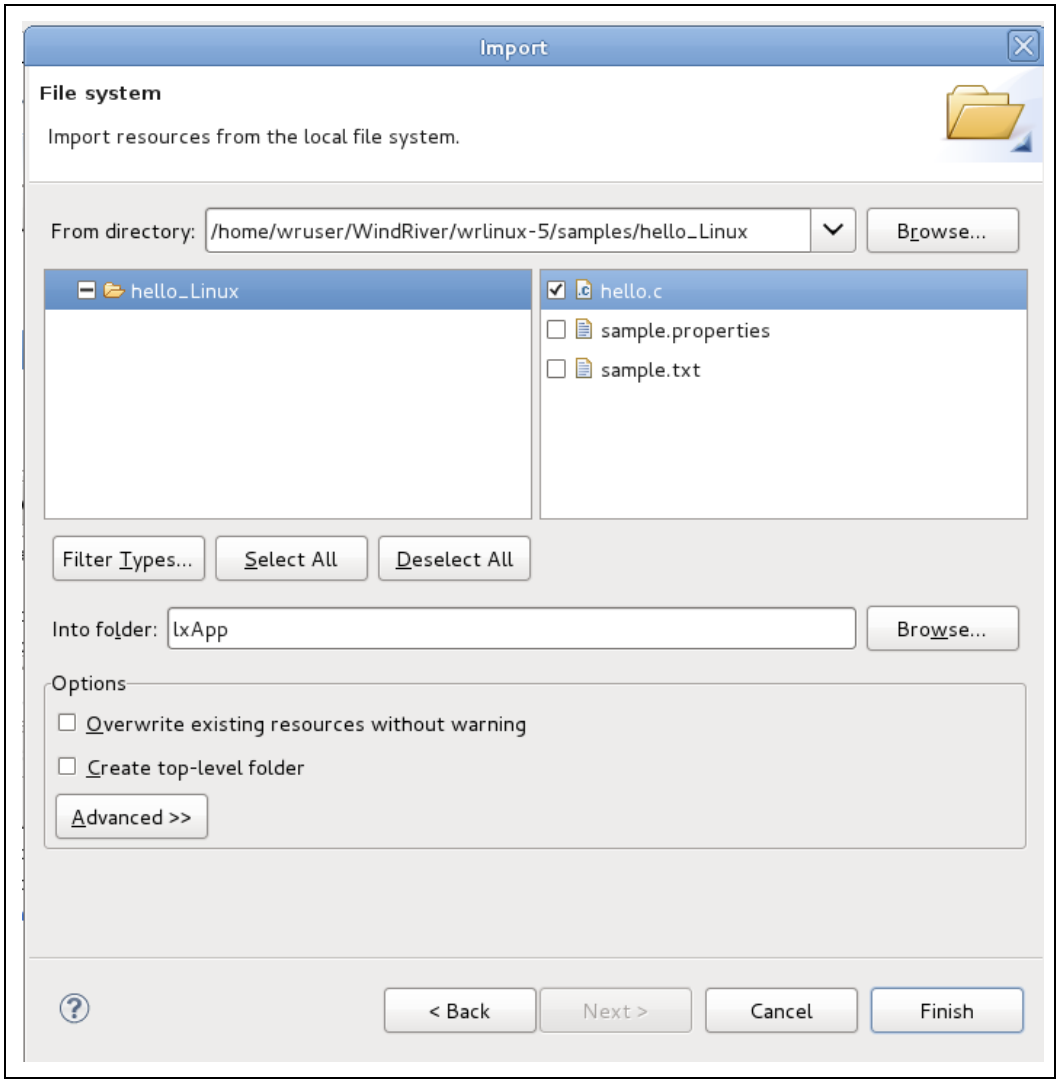
Browse to the following directory containing the application “hello.c”.

/home/wruser/WindRiver/wrlinux-5/samples/hello_Linux

Click **OK**. The application directory folder icon appears in the left pane.

In the right pane, check the desired application icon(s) you wish to include in Wind River Linux application project. In this example, click **hello.c**. See Figure 109 - Import.

Figure 109 - Import



Click **Finish**. The window closes.

Right click **lxApp (Wind River Linux Platform Application Project)** and select **Build Project**.

NOTE: If a **Build** warning window (denoted by a light bulb symbol) appears prompting you to include the search path for your application, select the **Continue** button.

After a few moments, in the **Build Console** tab a message appears stating **Build Finished in 'lxApp'**

8.7 Build the Hypervisor Integration Project (HIP)

In this section, you modify the Wind River Hypervisor Integration Project (HIP). The HIP creates a bootable Hypervisor image, which you then load into a target as a embedded virtualization system. A HIP is most commonly used in a system integration scenario where you are configuring the Guest OSes and their resources.

We will cover how to take the two VxWorks images created in section 2 and the WR Linux image created in Section 3 and integrate them into the hypervisor.

8.7.1 Modifying the HIP

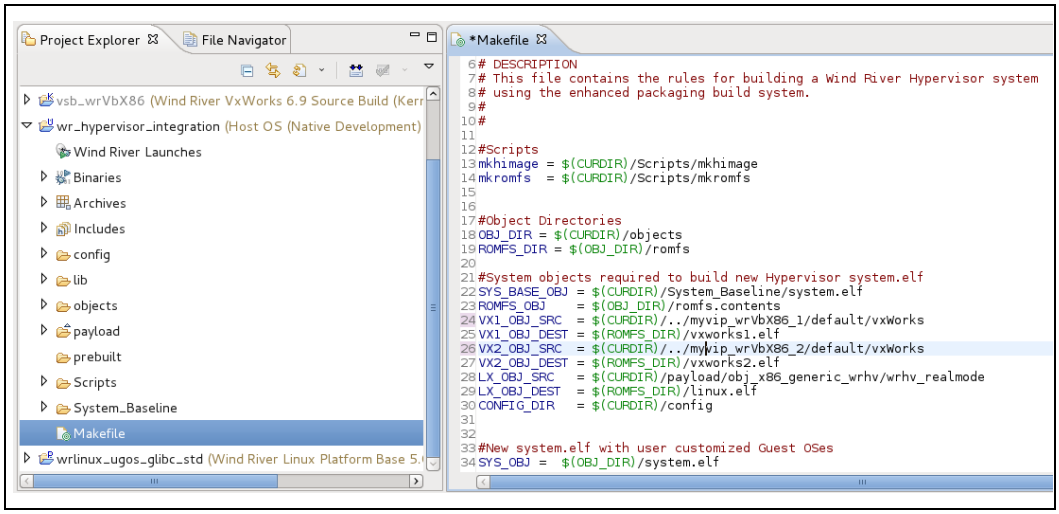
The Wind River Workbench environment comes with a pre-loaded HIP named wr_hypervisor_integration. You will use the existing project, and simply modify it to use the projects you created earlier.

Perform the following steps.

In Wind River Workbench, expand the **wr_hypervisor_integration** project and double click the **Makefile** to bring it up in the editor.

See Figure 110 – Hypervisor Updating.

Figure 110 – Hypervisor Updating



Edit the following two lines to point to the path of the projects created in the previous sections:

- Update **VX1_OBJ_SRC** to point to use **myvip_wrVbX86_1**
- Update **VX2_OBJ_SRC** to point to use **myvip_wrVbX86_2**

IMPORTANT: Make certain to update the path value with the entire path name.

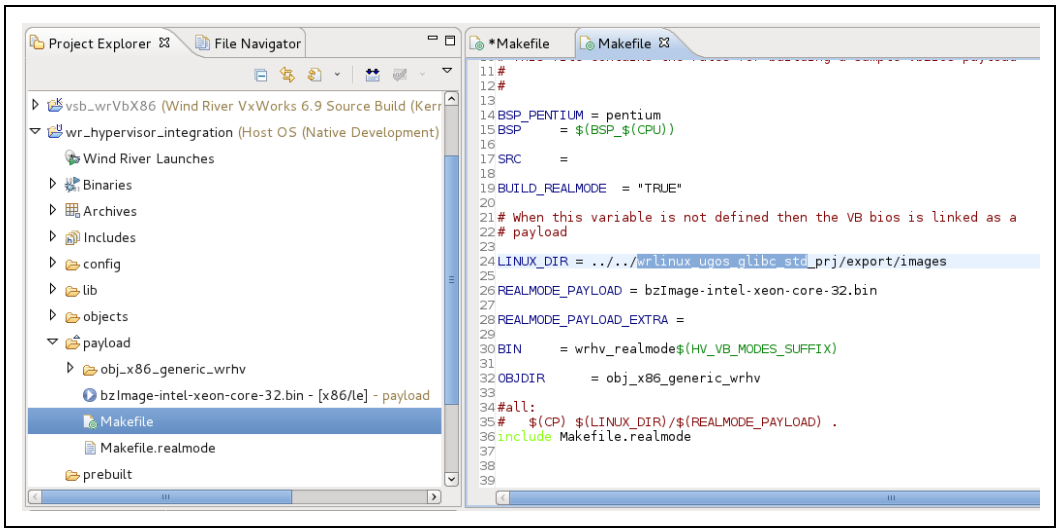
With these changes, the integration project automatically copies your VxWorks guest OS into the **wr_hypervisor_integration** project file system.

In the **wr_hypervisor_integration** project browse to the file **wr_hypervisor_integration > payload**

Double-click **Makefile**. A **Makefile** pane appears.

Verify that the **LINUX_DIR** path matches the Linux project name you use in the previous section. See Figure 111 – Project Name.

Figure 111 – Project Name



8.7.2 Build the Hypervisor Project

This section provides instruction for how to build the guest OSes into the Hypervisor kernel.

NOTE: More information about the Hypervisor build process can be found in the *Wind River Hypervisor Getting Started Guide 2.0, section 3.4* and the *Wind River Hypervisor User's Guide 2.0, section 12*.

To build Hypervisor, go to **Project Explorer** and right-click the **wr_hypervisor_integration** project.

Select **Rebuild Project.** A **Rebuild Project** window showing a progress bar appears. In the **Build Console** tab at the bottom of Workbench, lines of code scroll as the system processes the build.

The system generates the Hypervisor binary, **system.elf**, and places it into:

/home/wruser/workspace/wr_hypervisor_integration/objects

8.8 Copy SYSTEM.ELF to Target Platform USB Media

Intel® Industrial Solutions System Consolidation Series SCS comes with a GRUB2 bootable target USB drive. Use this USB drive to simplify the process of booting the system on the target.

Follow the procedure below to use the target platform USB media to boot the target platform:

Plug the Target USB into the development host.

The Target USB automatically mounts to **/media/<mount_dir>**

In the **Project Explorer** tab, right-click the **wr_hypervisor_integration** HIP and select **Open Wind River Workbench 3.3.4 Development Shell**. A command line interface window opens into the directory of the Hypervisor Integration Project you created previously.

In this next step you back up the **system.elf** file that currently resides in the **/images** directory on the USB.

NOTE: The file paths given in the command-line steps below are for example purposes only. The file path on which your **system.elf** file resides may vary. To determine the exact path of your **system.elf** file, in Project Explorer right-click the **system.elf** object and select **Properties**. A window appears showing the file path.

At the command line interface, type the command below and press **Enter**.

```
cp /media/<mount_dir>/images/system.elf
/media/<mount_dir>/images/system.elf.backup
```

Copy the new ELF file to the **images** directory on the USB. Into the command line interface, type the command below and press **Enter**:

```
cp /home/wruser/workspace/wr_hypervisor_integration/objects/
/system.elf /media/<mount_dir>/images/system.elf
```

Alternatively use the development host Linux File Explorer to copy the **system.elf** file.

Unmount and remove the target USB from the development host.

Plug the Target USB media into the development host.

The Target USB automatically mounts to **/media/<mount_dir>**

In the **Project Explorer** tab, right-click the **wr_hypervisor_integration** project and select **Open Wind River Workbench 3.3.3 Development Shell**. A command line interface window opens into the directory of the Project.

Copy the new ELF file onto the USB. Into the command line interface, type the command below and press **Enter**:

```
cp objects/system.elf  /media/<mount_dir>/system.elf
```

Unmount and remove the target USB from the development host.

Power down the target platform.

8.9 Open a Development Host AMIO Connection

On the development host, establish an open physical serial connection.

On the development host Workbench, go to **Window > Show View > Other.** A **Show View** window appears.

Expand the **Wind River Workbench** option.

Select **AMIO Console** and click **OK**. The **Show View** window disappears and an **AMIO Console** tab appears in the Workbench bottom pane.

Click the **Open Icon** to the right of the **AMIO** tab that looks like multiple consoles overlaid upon each other. See Figure 112 – Open Icon. An **Open AMIO Console** window appears.

Figure 112 – Open Icon

In the **Open AMIO Console** window, for the **Connection Type** select **Serial Port**.

At the **Port:** dropdown list select the host serial port to which the target is connected.

From the **Baud Rate:** dropdown list select 9600.

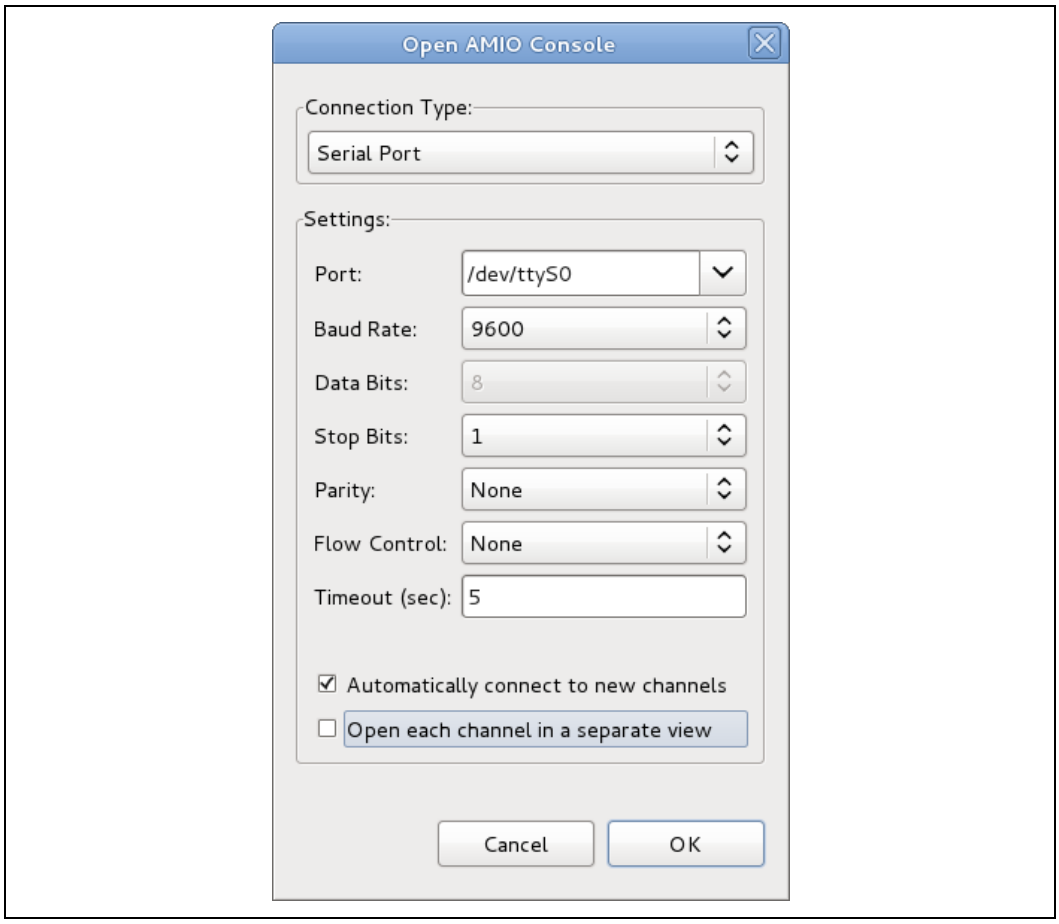
For the remaining terminal settings, enter the values shown below:

- Data Bits: 8
- Stop Bits: 1
- Parity: None
- Flow Control: None
- Timeout (sec): 5

Check the **Automatically connect to new channels** option.

Uncheck the **Open each channel in a separate view** option. See Figure 113 - Channel.

Figure 113 - Channel



Click **OK**. The **Open AMIO Console** window closes. At the bottom of Workbench, a new tab appears under the AMIO console.

8.10 Configure Target Platform BIOS to Boot from USB Media

Before booting the target platform from USB media, you first configure the target platform BIOS to boot from USB media.

NOTE: To ensure that you start the target platform from USB media, follow this procedure each time you need to boot from USB media.

Shut down the target platform and wait 5 seconds.

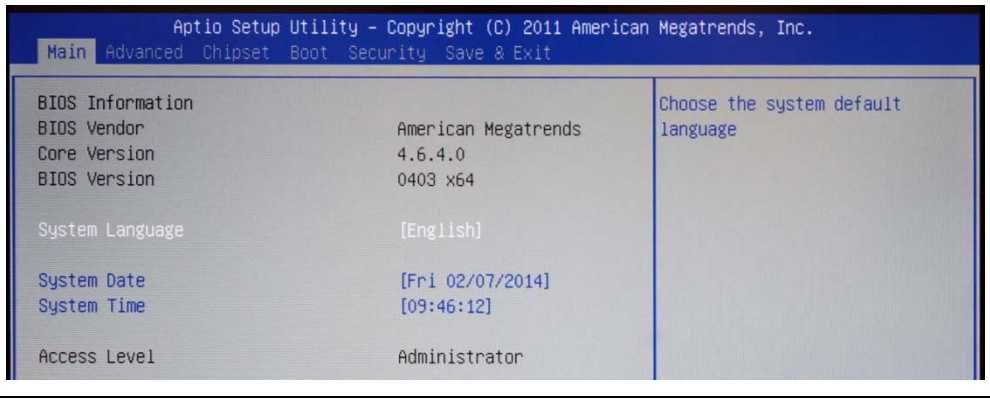
Place the bootable USB media into an open USB port (preferably a blue USB 3.0 port).

Press and release the target platform power button. The target platform begins to start.

As the target platform begins to start, repeatedly press the **DELETE** button.

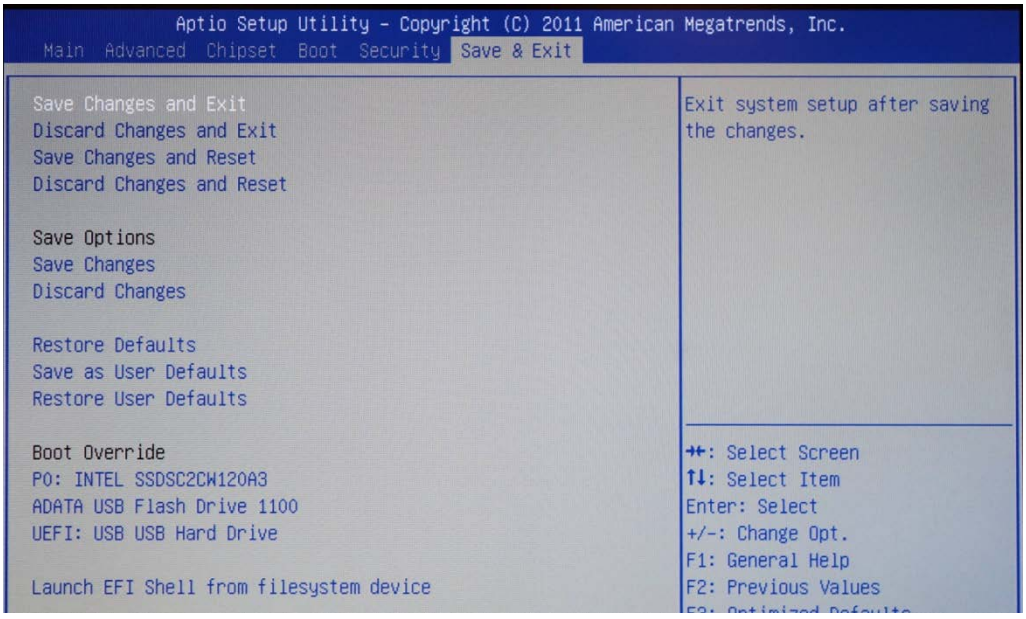
The target platform enters the BIOS **Main** tab. See Figure 114 – BIOS 1.

Figure 114 – BIOS 1



Use the right-arrow key to go to the **Save & Exit** tab. See Figure 115 – BIOS 2.

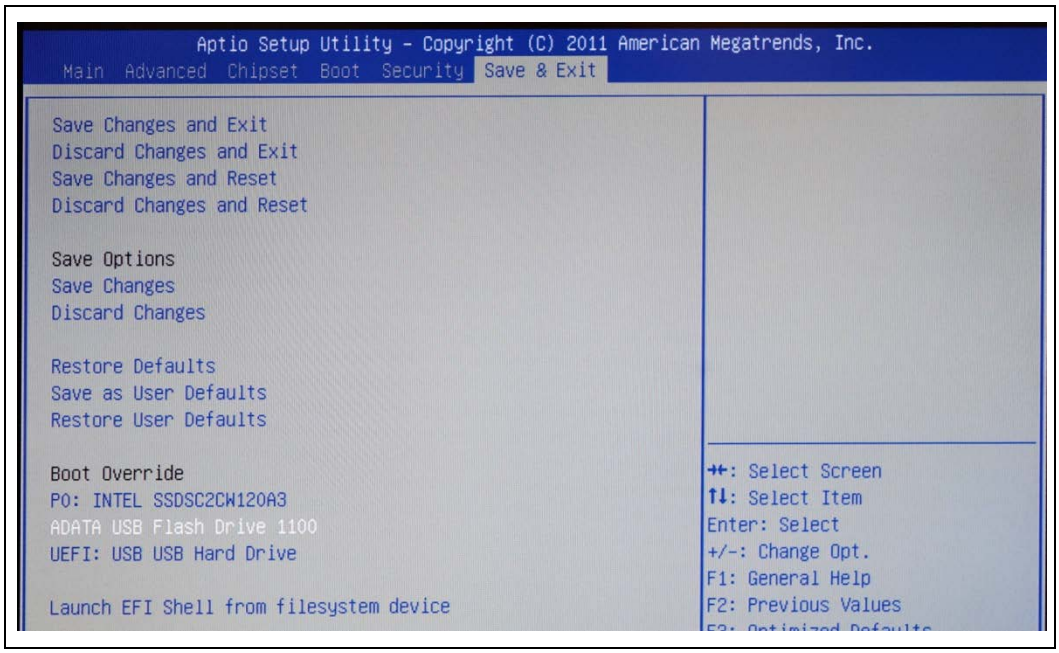
Figure 115 – BIOS 2



Use the down-arrow key to move the highlighted selection to the USB media. See Figure 116 – BIOS 3.

NOTE: In this example, the USB media is named **ADATA USB Flash Drive 1100**. Your USB media may have a different name.

Figure 116 – BIOS 3



Press **Enter**. The target platform boots from USB media.

As the system starts, communication activity appears on the development host AMIO console. Several console tabs appear within the AMIO pane:

- **Core OS –** This shows the Hypervisor I/O
- **Channel 1 –**This shows the Linux I/O
- **Channel 2 –** This shows the I/O for VxWorks #1
- **Channel 3 –** This shows the I/O for VxWorks #2

8.11 Confirm Sample Code Functionality

Follow this procedure to confirm the functionality of the sample code items you embedded into the VxWorks 1 and VxWorks 2 OSes.

8.11.1 Verify DKM Functionality In VxWorks 1 OS

Follow this procedure to verify that the sample code embedded into the VxWorks 1 OS functions.

After the target platform boots, go to AMIO – Channel 2 for VxWorks 1 OS.

At the command prompt type the name of the executable application.

```
-> testFactory
```

Press **Enter**. Output appears below the command. See Figure 117 - VxWorks 1 Code.

Figure 117 - VxWorks 1 Code

```
-> testFactory
classRegistryShow ()
Showing Class Registry ...
Name Address
=========================================================
blue_t  0xeb7190
green_t         0xeb71e8
red_t   0xe95940
objectRegistryShow ()
Showing Object Registry ...
Name    Address
=========================================================
objectCreate ("green_t", "bob")
Creating an object called 'bob' of type 'green_t'
objectCreate ("red_t", "bill")
Creating an object called 'bill' of type 'red_t'
objectRegistryShow ()
Showing Object Registry ...
Name    Address
=========================================================
bill    0xeb74c0
bob     0xeb7468
objectTypeShowByName ("bob")
Looking up object 'bob'
Attempting to ascertain type of object at 0xeb7468
Attempting a dynamic_cast to red_t ...
dynamic_cast threw an exception ... caught here!
Attempting a dynamic_cast to blue_t ...
dynamic_cast threw an exception ... caught here!
Attempting a dynamic_cast to green_t ...
Cast to green_t succeeded!
green.
value = 7088960 = 0x6c2b40 = '@' = std::cout
->
```

Presence of this content signifies the application is present in the OS and is working correctly.

NOTE: If the application does not exist in the booted target platform, then the target platform system most likely is not booting from the USB media, and is booting from the target platform SSD instead. See Section 8.10 – Configure Target Platform BIOS to Boot from USB Media for instructions to configure the target platform to boot from the USB media.

8.11.2 Verify RTP Functionality In VxWorks 2 OS

Follow this procedure to verify that the sample code embedded into the VxWorks 2 OS functions.

After the target platform boots, go to AMIO – Channel 3 for VxWorks 2.

NOTE: For the steps that follow, refer to the output content shown in Figure 118 - VxWorks 2 Code to verify that you have performed the procedure correctly.

Figure 118 - VxWorks 2 Code

```
->
-> cmd
[vxWorks *]#
[vxWorks *]# cd /romfs
[vxWorks *]# ls
.
..
helloworld.vxe
[vxWorks *]# helloworld.vxe
Launching process 'helloworld.vxe' ...
Process 'helloworld.vxe' (process Id = 0x11718c0) launched.
Hello World
[vxWorks *]#
```

Type **cmd** and press **Enter**. The bracketed **cmd** prompt appears.

Type **cd /romfs** and press **Enter**.

Type **ls** and press **Enter** to list files within directory. The executable **helloworld.vxe** should appear in the directory list.

Type **helloworld.vxe** and press **Enter.**

The application should launch. The output is **Hello World**.

This output signifies the application is present in the OS and is working correctly.

NOTE: If the application does not exist in the booted target platform, then the target platform system most likely is not booting from the USB media, and is booting from the target platform SSD instead. See Section 8.10 – Configure Target Platform BIOS to Boot from USB Media for instructions to configure the target platform to boot from the USB media.

§

9 Workflow 4: Dynamically Load & Debug Applications on the Target Platform

9.1 About This Chapter

The Intel® Industrial Solutions System Consolidation Series development host uses *Wind River* Workbench* as its primary development tool. Use the Workbench development platform for innumerable development activities called *workflows*.

IMPORTANT: Before you perform this workflow, you must perform Workflow 1. Intel Corporation also recommends performing all other workflows previous to this workflow in the order they occur in this document.

This chapter includes both system *background information* and *instructions* for exploring system features.

- FIRST… Read and understand this chapter entirely.
- THEN… Re-read the chapter and perform the instructions in the workflow.

After performing the other workflows in this document, you find that one of your applications does not function correctly or requires modification. Instead of rebuilding the entire Hypervisor system and rebooting the target platform, during development you can build and dynamically load just the application to the target platform. In this workflow you will have the opportunity to learn how to port your own source code to both Linux* and the VxWorks* partitions.

9.2 Initialize Linux User Mode Agent

By default, Linux's debug agent does not execute at start-up. In order to begin a remote session with the Linux OS, start the User Mode Agent. Follow the procedure.

Connect the target platform to an Ethernet cable before booting up.

Use Workbench to connect to the booted target platform.

Go to the Workbench Linux AMIO console.

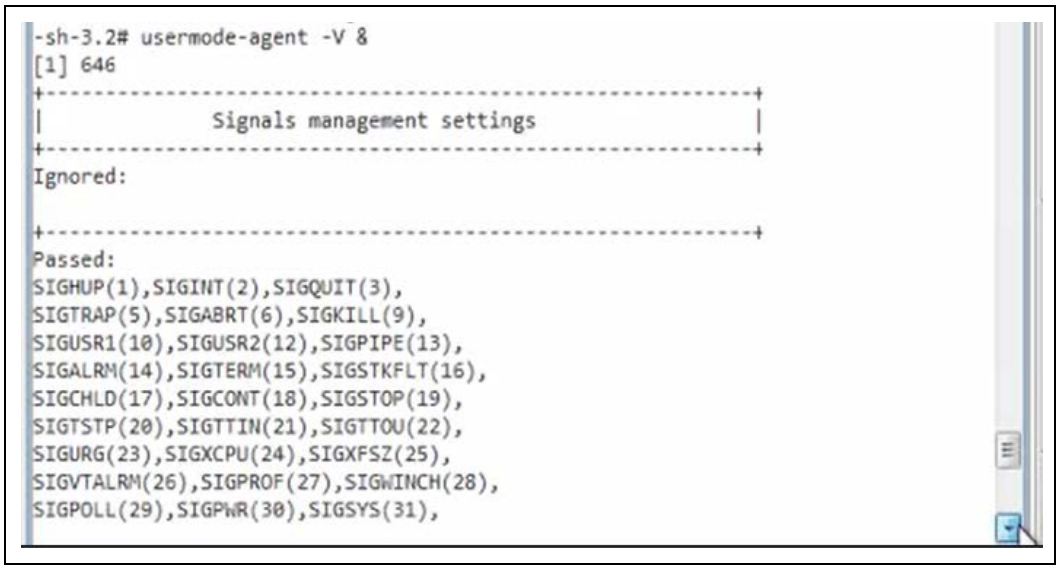
At the Linux command prompt type

```
usermode-agent -V &
```

Press **Enter**.

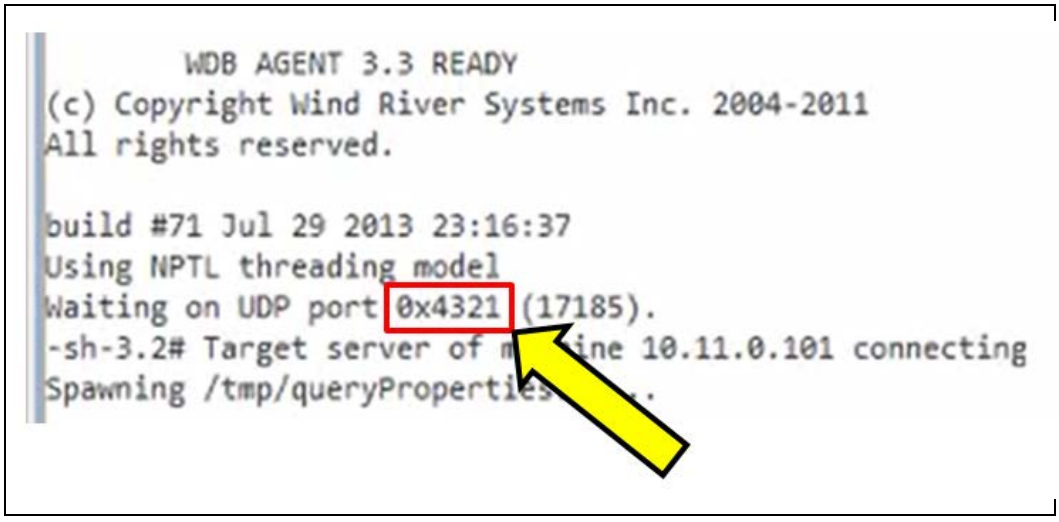
The console produces many lines of output. Figure 119 - Usermode Agent Output displays the top portion of this output.

Figure 119 - Usermode Agent Output



Scroll to the section of output that displays the UDP port number that the agent monitors. See the boxed item in Figure 120 - Agent Monitor Port.

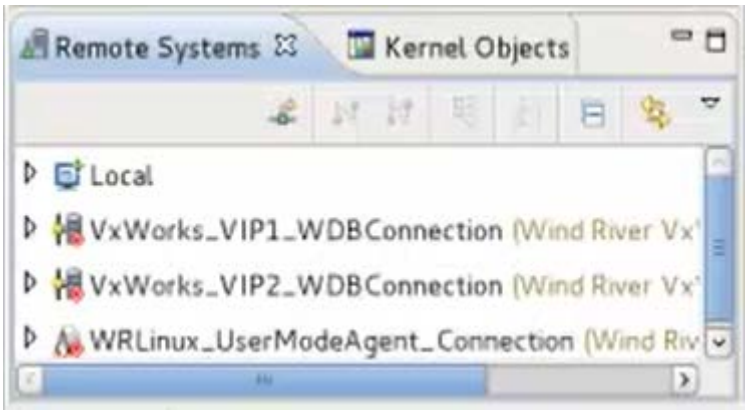
Figure 120 - Agent Monitor Port


WDB AGENT 3.3 READY
(c) Copyright Wind River Systems Inc. 2004-2011
All rights reserved.
build #71 Jul 29 2013 23:16:37
Using NPTL threading model
Waiting on UDP port 0x4321 (17185).
-sh-3.2# Target server of
ine 10.11.0.101 connecting
Spawning /tmp/queryProperties

Record the port number.

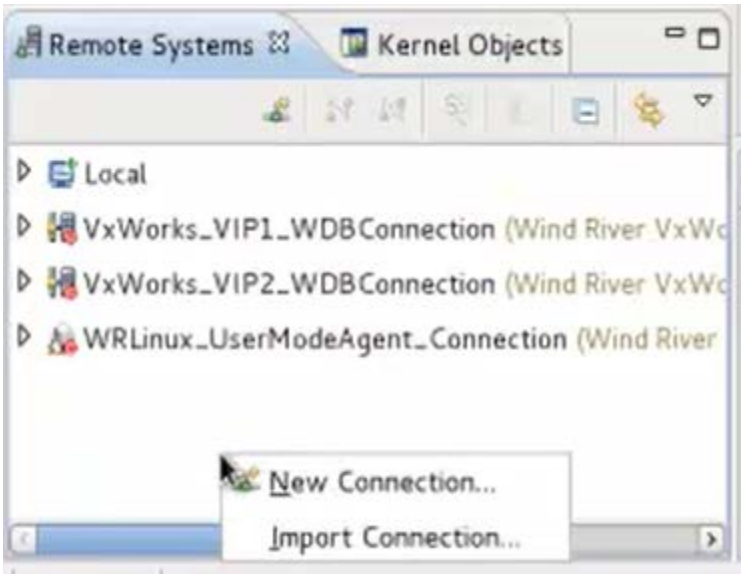
Go to the Workbench **Remote Systems** pane. See Figure 121 – Remote Systems Window.

Figure 121 – Remote Systems Window



Right-click the white space within the **Remote Systems** window. A menu appears. See Figure 122 – White Space.

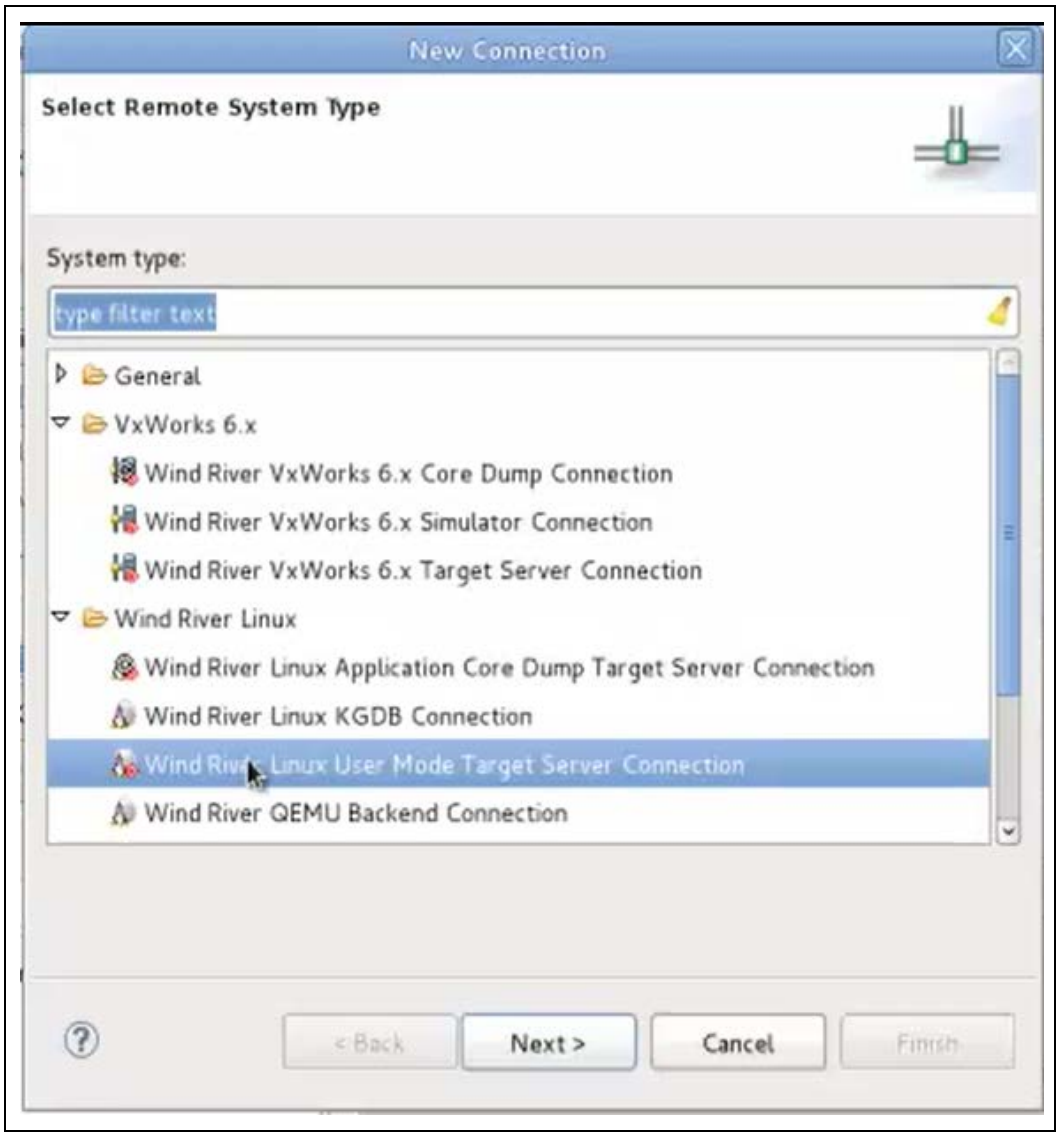
Figure 122 – White Space



Select **New Connection**. A **Select Remote System Type** window appears.

Select Wind River Linux User Mode Target Server Connection. See Figure 123 – System Type.

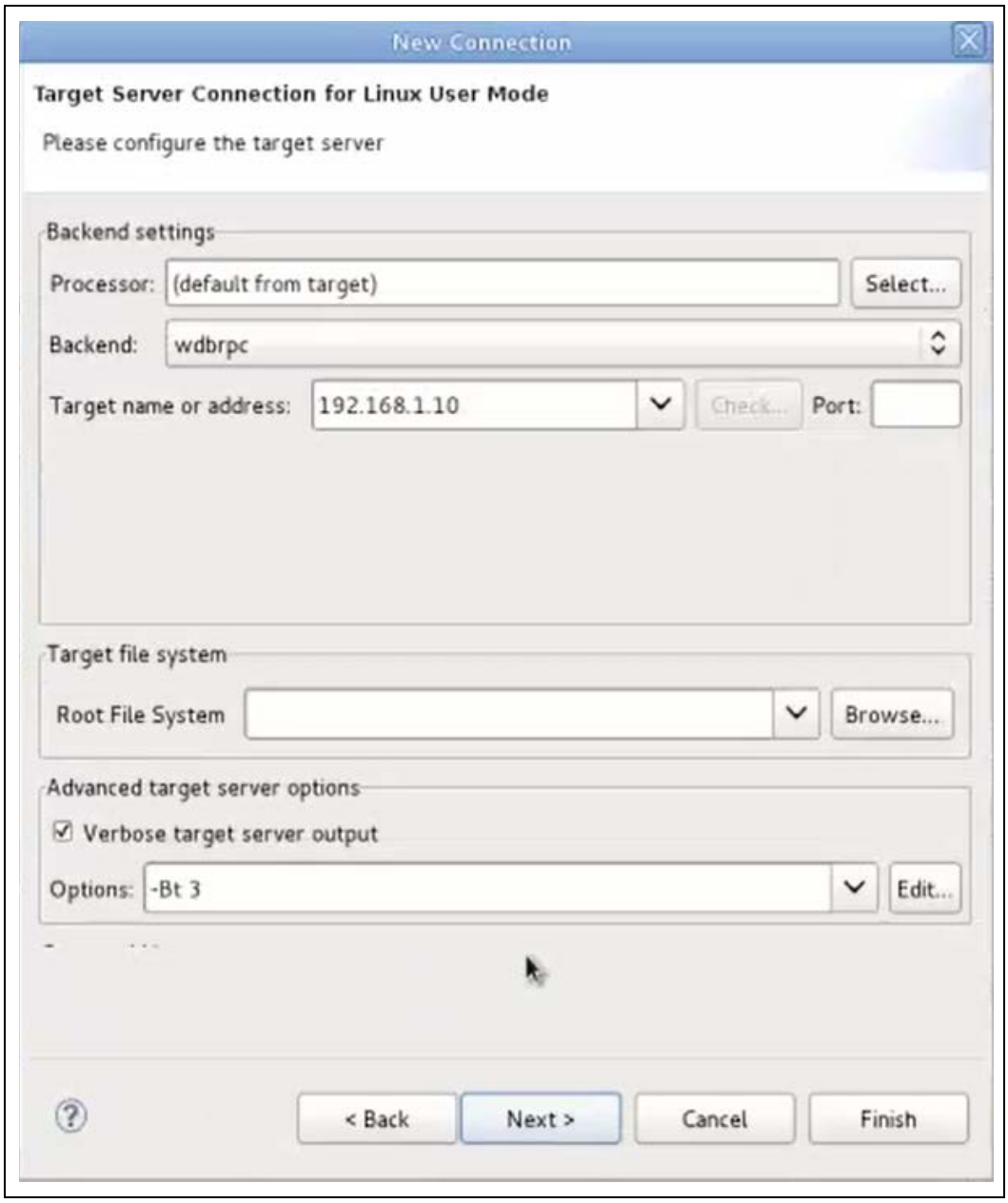
Figure 123 – System Type


New Connection
Select Remote System Type
System type:
type filter text
General
VxWorks 6.x
Wind River VxWorks 6.x Core Dump Connection
Wind River VxWorks 6.x Simulator Connection
Wind River VxWorks 6.x Target Server Connection
Wind River Linux
Wind River Linux Application Core Dump Target Server Connection
Wind River Linux KGDB Connection
Wind River Linux User Mode Target Server Connection
Wind River QEMU Backend Connection
< Back
Next >
Cancel
Finish

Click **Next**. A **Target Server Connection for Linux User Mode** window appears.

Into the **Target name or address:** field, enter the Linux OS IP address. Figure 124 – Target Server…. uses **192.168.1.10** as the IP address.

Figure 124 – Target Server…



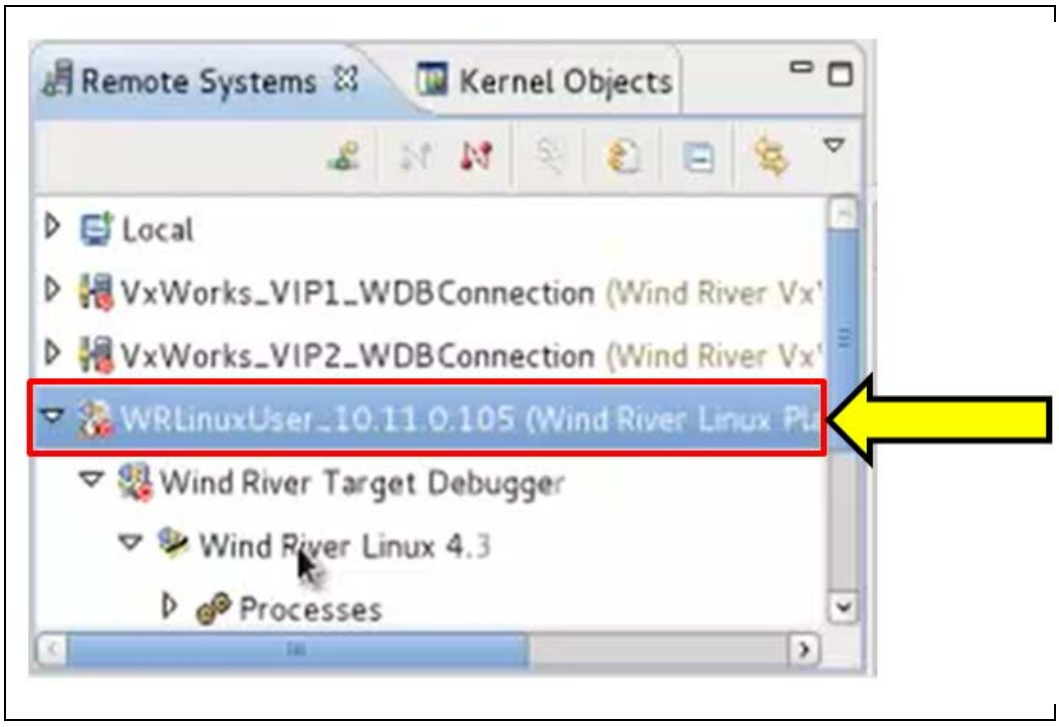
NOTE: For more information about identifying or configuring the IP addresses for target platform OSes, see Section 10 – Workflow 5: Assign Persistent Target Platform OS IP Addresses.

Into the **Port:** field type **0x** (zero and x) and the port number you recorded previously. For example if the number you recorded previously was 4321, type **0x4321**.

Into the **Root File System** field enter **/bin**

Click **Finish**. The development host connects to the target platform and a new connection icon appears in the **Remote Systems** window. A remote connection is established. See the arrowed item in Figure 125 – Remote Connection.

Figure 125 – Remote Connection



Expand the icons under the new connection until you see the **Processes** object. See the **Processes** object in Figure 125 – Remote Connection.

Optionally expand the **Processes** icon to view the various OS processes.

9.3 Create a Linux User Application

Go to **File → New → Project**. A new Project window opens.

Go to the **Wind River Linux** icon and expand it. The icon options expand beneath.

Select **Wind River Linux Application Project**.

Click **Next**. A **Project** window appears.

In the **Project Name:** field type **wrLxProject**

Click **Finish**. A **wrLxProject (Wind River Linux Platform Application Project)** object appears in **Project Explorer**.

In **Project Explorer**, right-click **wrLxProject (Wind River Linux Platform Application Project)** and select **Import**. A **Select** window appears.

In the **Select an import source:** list, expand the **General** icon. The icon options expand beneath.

Select **File System** and click **Next**. An **File System** window opens.

Adjacent to the **From directory:** field, click **Browse…** and browse to your application directory.

NOTE: In the following step this workflow example imports an application called “hello.c”. To import your own code, substitute the directory path below with the directory to your own source code.

Browse to the following directory containing the application “hello.c”.

/home/wruser/WindRiver/wrlinux-5/samples/hello_Linux

Click **OK**. The application directory folder icon appears in the left pane.

In the right pane, select the desired application icon(s) you wish to include with Wind River Linux.

Click **Finish**. The window closes.

Right click **wrLxProject (Wind River Linux Platform Application Project)** and select **Build Project**.

NOTE: If a **Build** warning window (denoted by a light bulb symbol) appears prompting you to include the search path for your application, select the **Continue** button.

After a few moments, in the **Build Console** tab a message appears stating **Build Finished in ‘wrLxProject’**

9.4 Download Application to a Linux Platform OS

Follow this procedure to download executables to a specific directory on the target platform Linux OS.

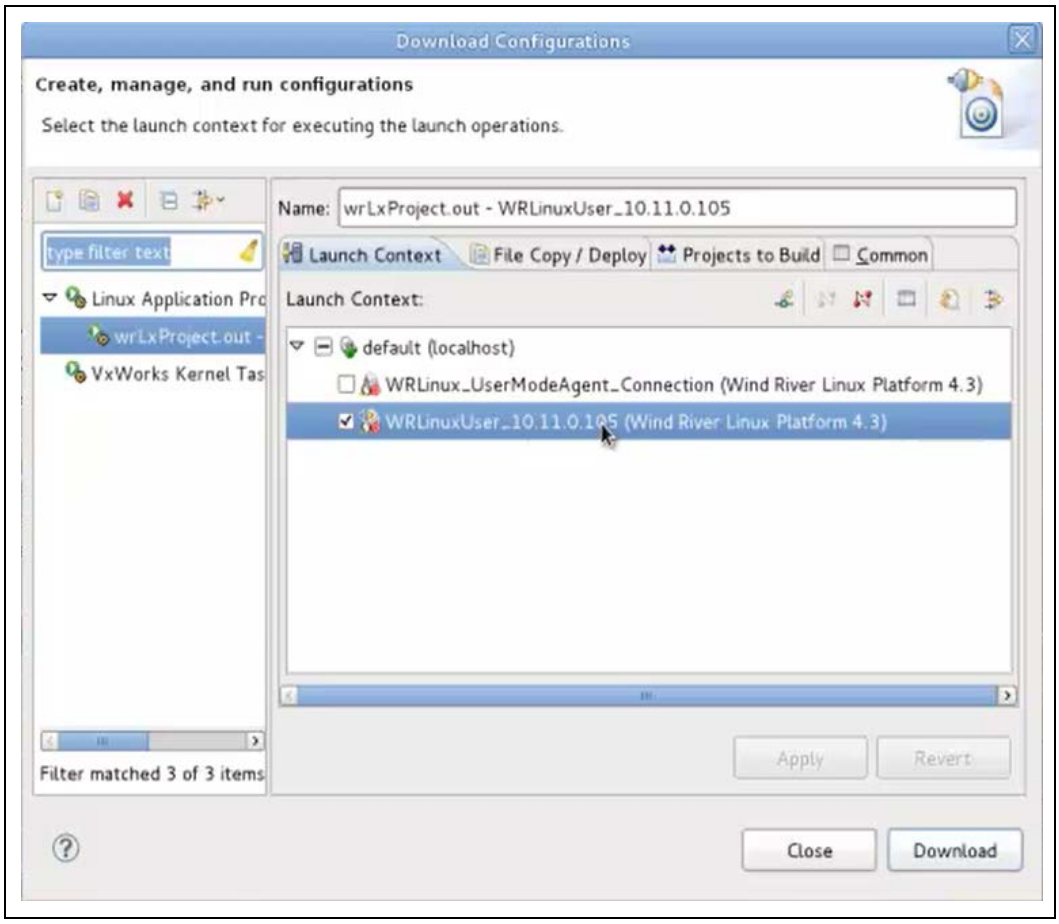
NOTE: In this case “download” means a real-time migration of content from the development host to the running target platform.

In Workbench Project Explorer, right-click **wrLxProject** and select **Download Linux Application Process**

In the **Create, manage, and run configurations** window, go to **Linux Applications Process** > **wrLxProject.out**.

In the **Launch Context:** tab, select the checkbox representing the target platform connection. See the checked item in Figure 126 – Download Configuration.

Figure 126 – Download Configuration



Click the **File Copy / Deploy** tab.

Use the bottom (horizontal) scroll bar to view the entire contents of the **File Copy / Deploy** window.

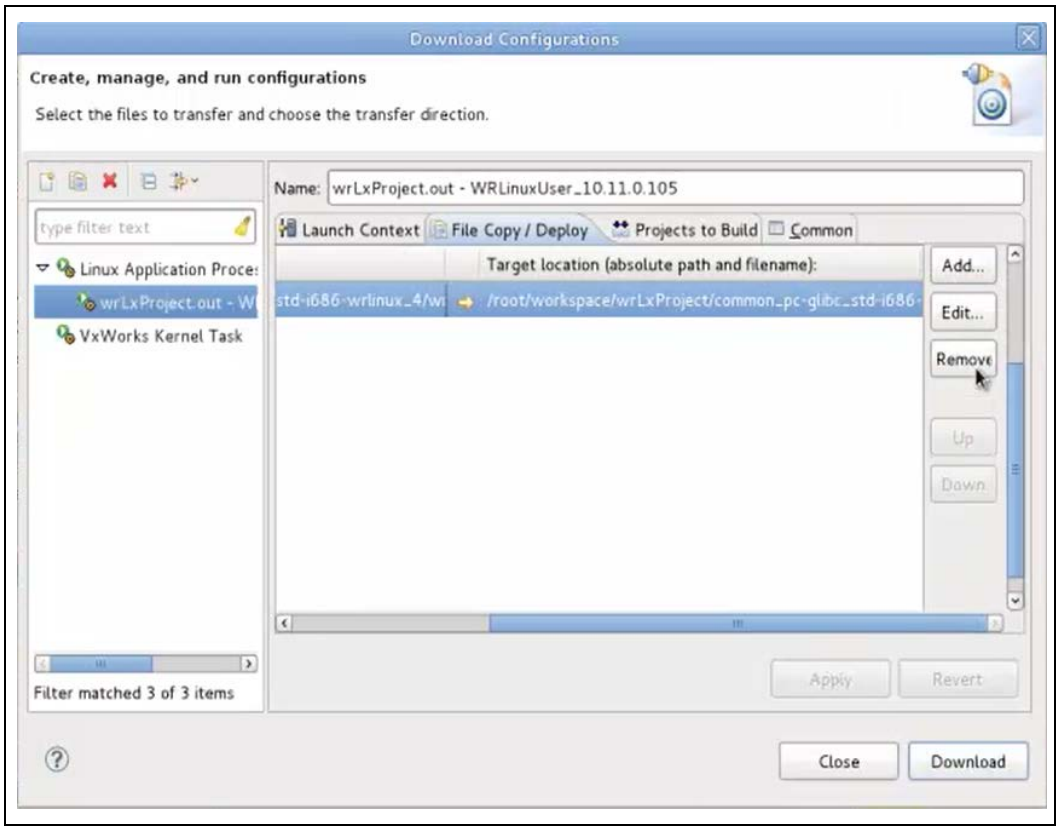
The window's development **Host location** column shows the location of the content you wish to deploy to the target. The executable is preconfigured since we are doing a download from an application project. If you have an existing executable that is not part of a workbench project, you could use the "Edit" button to replace the file, or even use "Add…" to download multiple applications.

The **Target Location (absolute path and filename)** column shows where on the target the file would be copied. Note that workbench does not know the file structure of the target, so the path is more than likely wrong and will need to be edited.

NOTE: If you initially open the download window from something other than a Linux Application project, it is possible that no files are present in the File Copy/Deploy tab. In that case, follow the instructions below to **Add…** a new download configuration.

Use the bottom (horizontal) scroll bar to view the right side of the **File Copy / Deploy** window. See Figure 127 – Edit File Copy/Deploy.

Figure 127 – Edit File Copy / Deploy



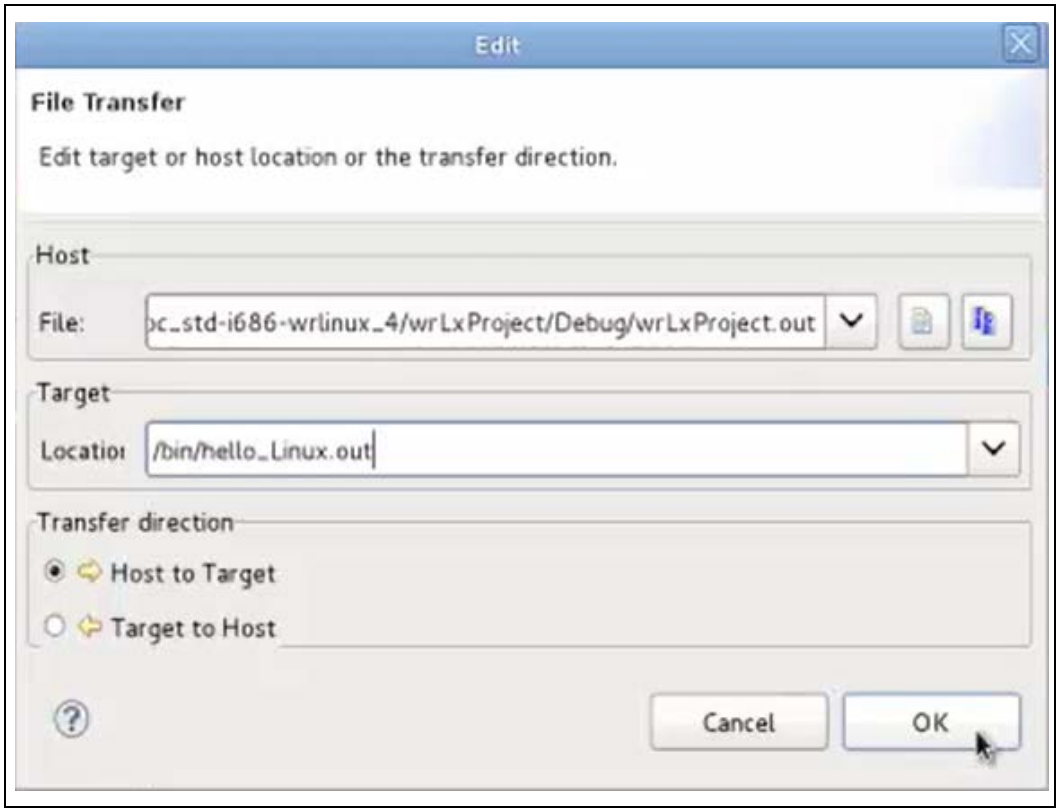
This movement shows the right side of the **File Copy / Deploy** window and three buttons:

- **Add...** – Click this button to create additional executable to deploy. You can use the line item for copying content from one location to another.
- **Edit...** – To use this function, within the **File Copy / Deploy** window select an existing download configuration and click the **Edit...** button. This action opens an **Edit** window where you view and change the development host and target platform file system locations for content you wish to copy.
- **Remove** – To use this function, select an existing download configuration. Click the **Remove** button to permanently remove that download configuration.

This procedure assumes you will edit and run an existing line item.

Click the **Edit...** button. An **Edit** window appears. See Figure 128 – Edit Window.

Figure 128 – Edit Window



Edit the locations of the host **File:** and target **Location:** fields if needed.

NOTE: Because the default path on the configuration usually refers to a path that does not exist on the target platform, you may need to edit the **Target Location** value.

Select the **Host to Target** radio button.

Click **OK**. The system copies the contents from the development host to the target platform. The **Edit** window closes and the **Download Configurations** window reappears listing the modified download configuration.

Click **Download**.

The system downloads the contents from the development host file location to the target platform target location. As the content transfers from host to target, on the bottom right of Workbench a console appears window showing the progress of the download and the content file structure build.

- If the download completes correctly, the console displays a **Build Finished** statement at the bottom of the console.

- If the download does not complete correctly, the console displays an error statement at the bottom of the console, with an indication of what the error entails.

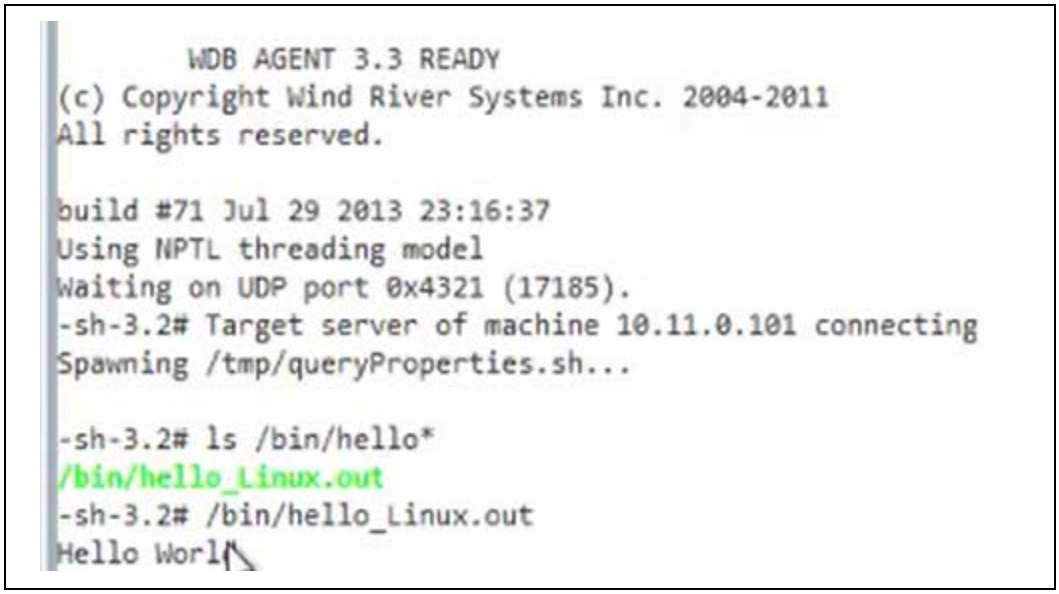
NOTE: This procedure assumes the download completes without errors.

In the Workbench AMIO console for Linux, navigate to the target directory where the executable content you downloaded resides, usually at:

/bin/hello_Linux.out

NOTE: This procedure uses an example called **hello_Linux.out**, which displays "Hello World" when run. See Figure 129 – AMIO Console.

Figure 129 – AMIO Console



Run the content and view the results.

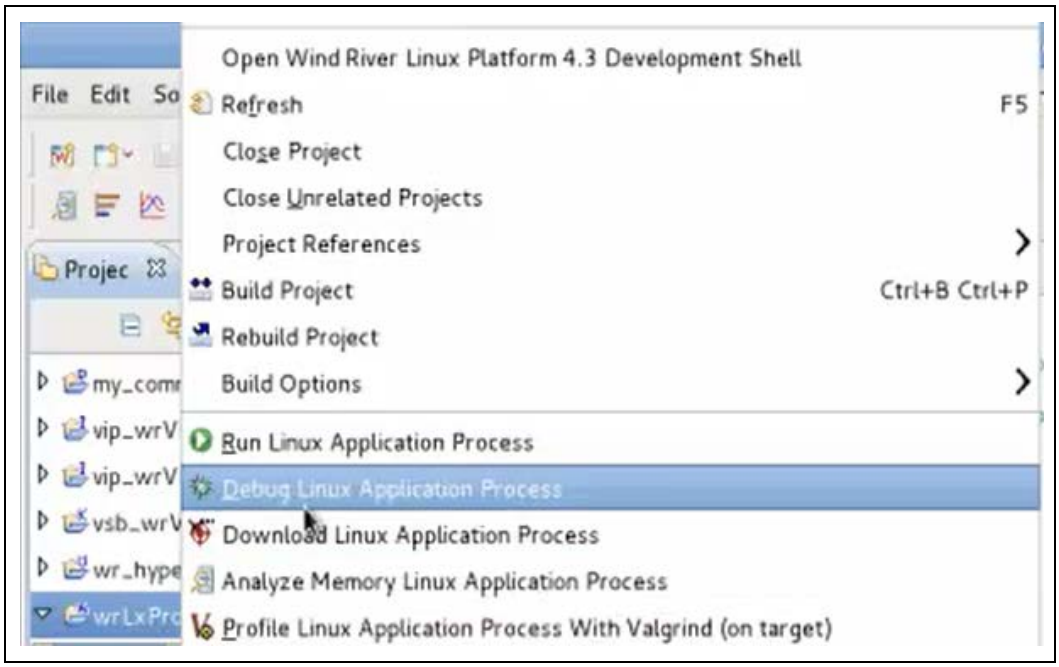
9.5 Debug Linux Target Platform Content

Use these steps to perform a debug procedure for a simple executable content piece in Linux. We use the project created in Section 9.3 – Create a Linux User Application.

In Workbench Project Explorer, select the **wrLxProject** project.

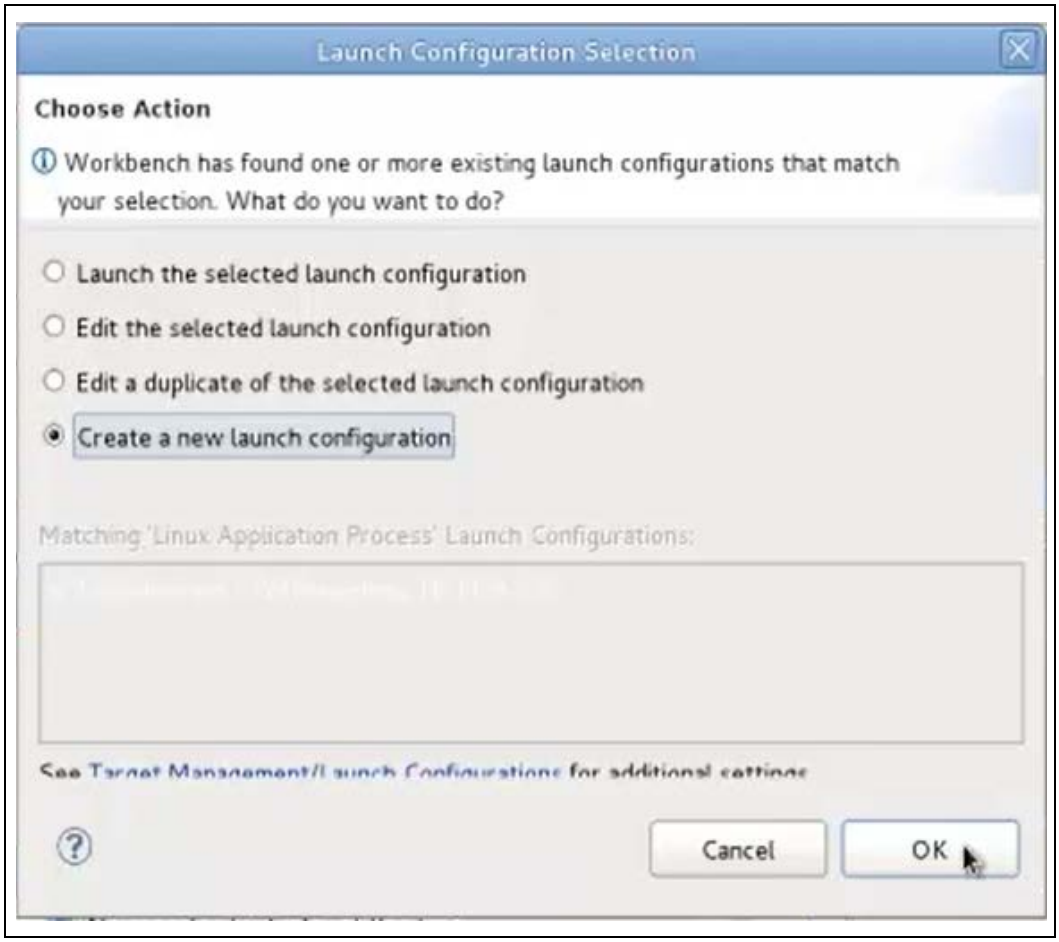
Right-click the Linux project and select **Debug Linux Application Process**. See Figure 130 – Debug Menu Item.

Figure 130 – Debug Menu Item



A **Choose Action** window appears. See Figure 131 – Choose Action Window.

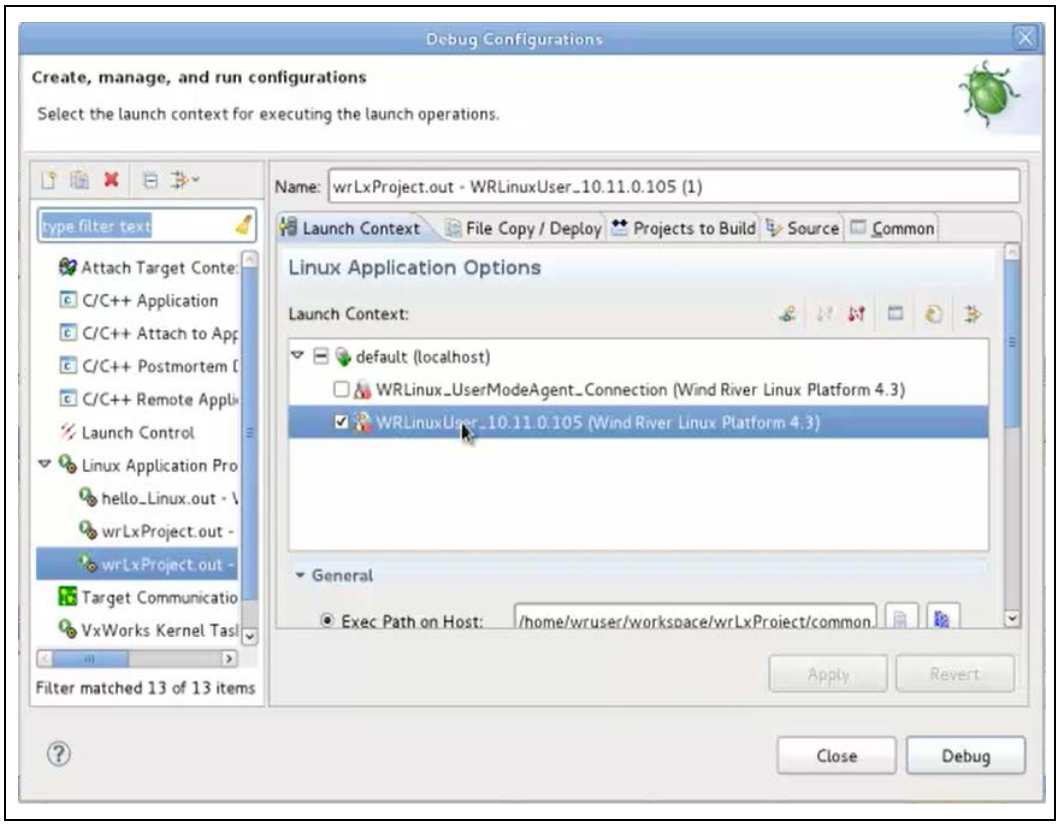
Figure 131 – Choose Action Window


Launch Configuration Selection
Choose Action
Workbench has found one or more existing launch configurations that match your selection. What do you want to do?
Launch the selected launch configuration
Edit the selected launch configuration
Edit a duplicate of the selected launch configuration
Create a new launch configuration
Matching 'Linux Application Process' Launch Configurations:
Cancel
OK

Select **Create a new launch configuration** and click **OK**. A Debug Configurations window appears. See Figure 132 – Create, Manage….

NOTE: This window appears almost identical to the **Edit File Copy / Deploy** window shown in Figure 127 – Edit File Copy/Deploy. See Figure 132 – Create, Manage….

Figure 132 – Create, Manage…



Select the target connection to use for debugging.

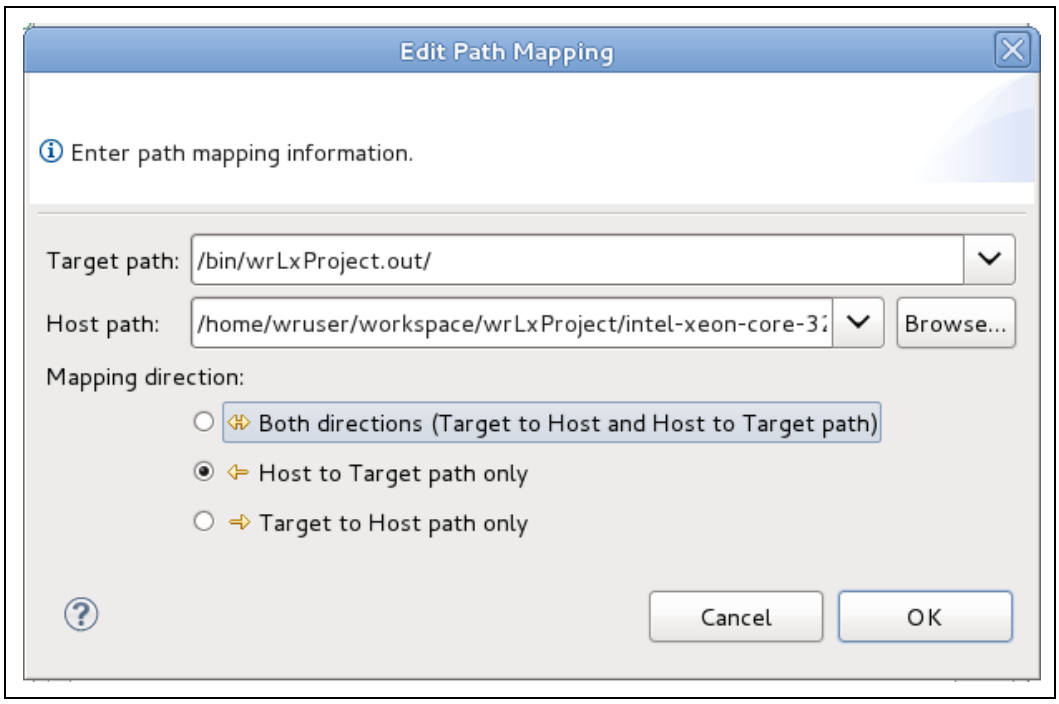
Click the **File Copy / Deploy** tab. The **File Copy / Deploy** window appears.

Use the bottom (horizontal) scroll bar to view the right side of the **File Copy / Deploy** window.

Optionally add, edit, or remove the connection as needed.

NOTE: Because the default path on the configuration usually refers to a path that does not exist on the target platform, you may need to edit the **Target Location** value. This procedure assumes you need to edit the debug configuration. See Figure 133 – Edit Window.

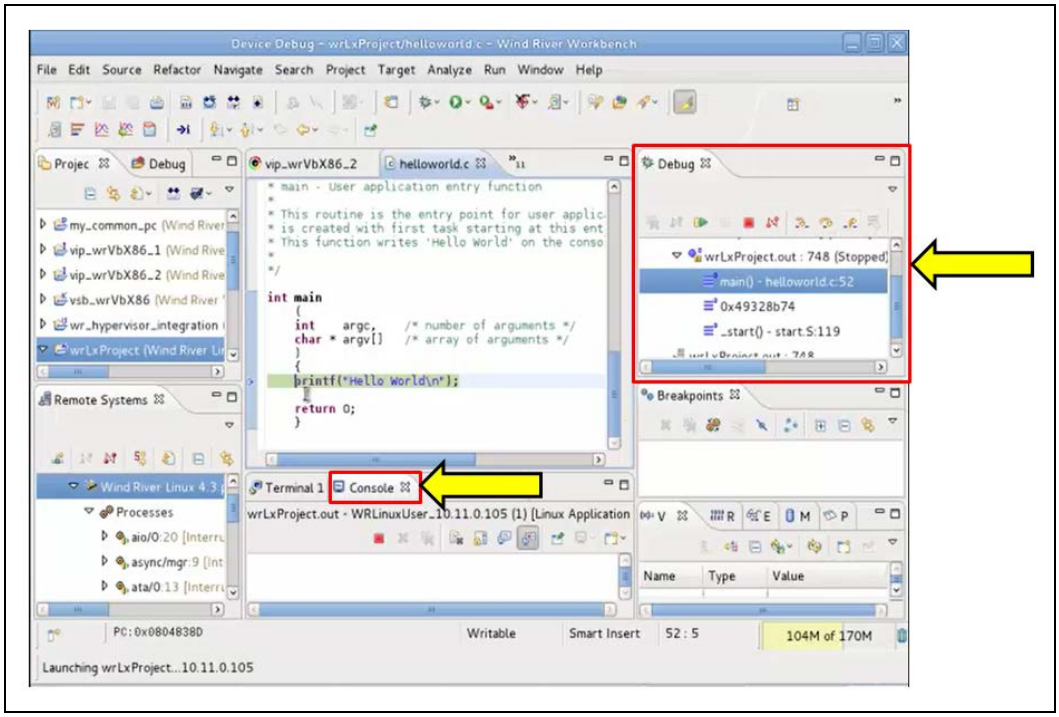
Figure 133 – Edit Window



When finished editing, click **OK**. The system copies the contents from the development host to the target platform. The **Edit** window closes and the **Download Configurations** window reappears listing the modified download configuration.

Click **Debug**. In Workbench a **Debug** console appears at the right, and an output **Console** appears at the bottom. See the two arrowed areas in Figure 134 – Debug and Console.

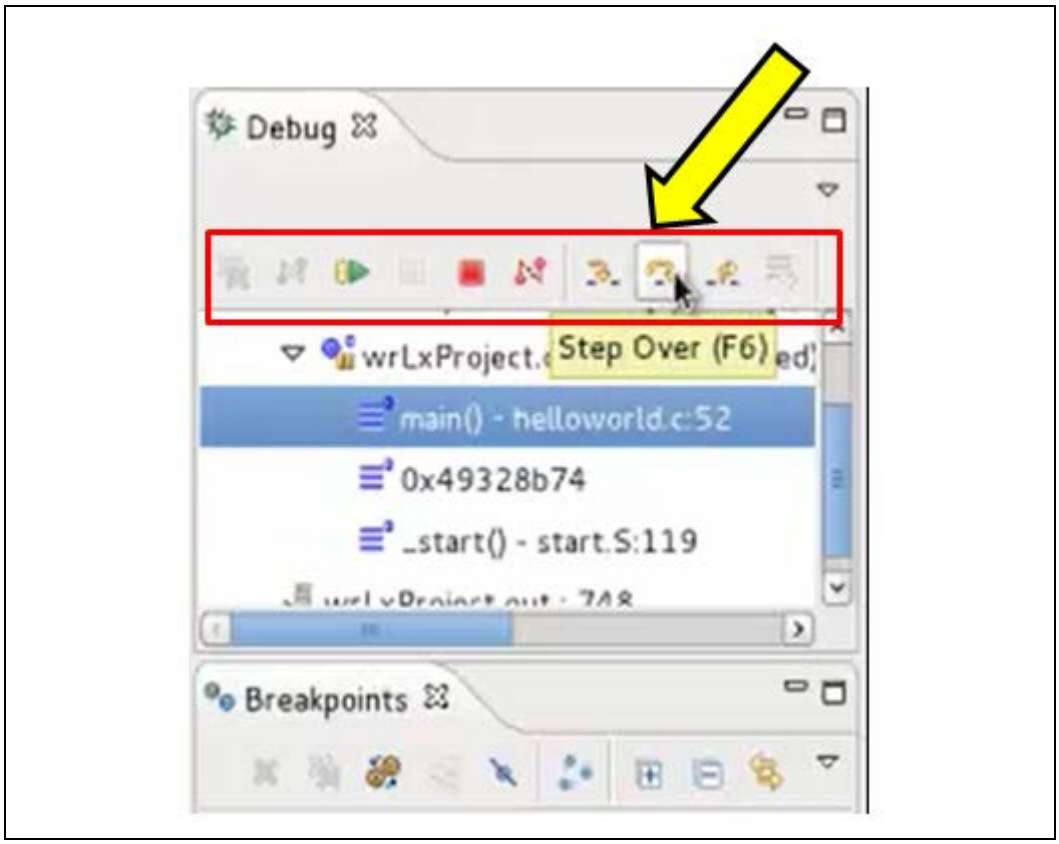
Figure 134 – Debug and Console



To discover their functions, hover the cursor over the various buttons in the **Debug** console. See the arrowed area in Figure 135 – Debug Menu Items.

NOTE: This procedure assumes you use the step over option in the debug process.

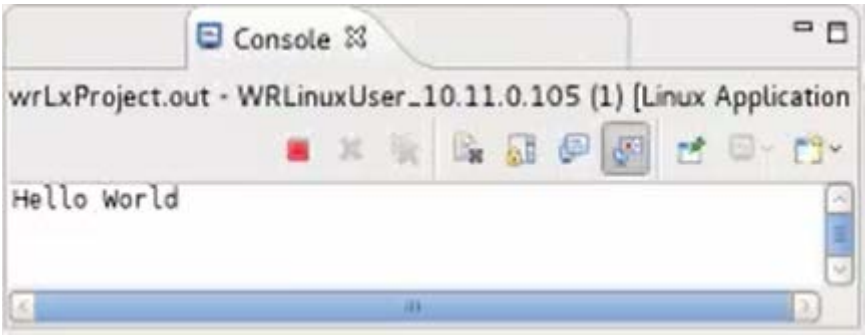
Figure 135 – Debug Menu Items



Click the **Step Over (F6)** icon to review the processes for the application you are debugging.

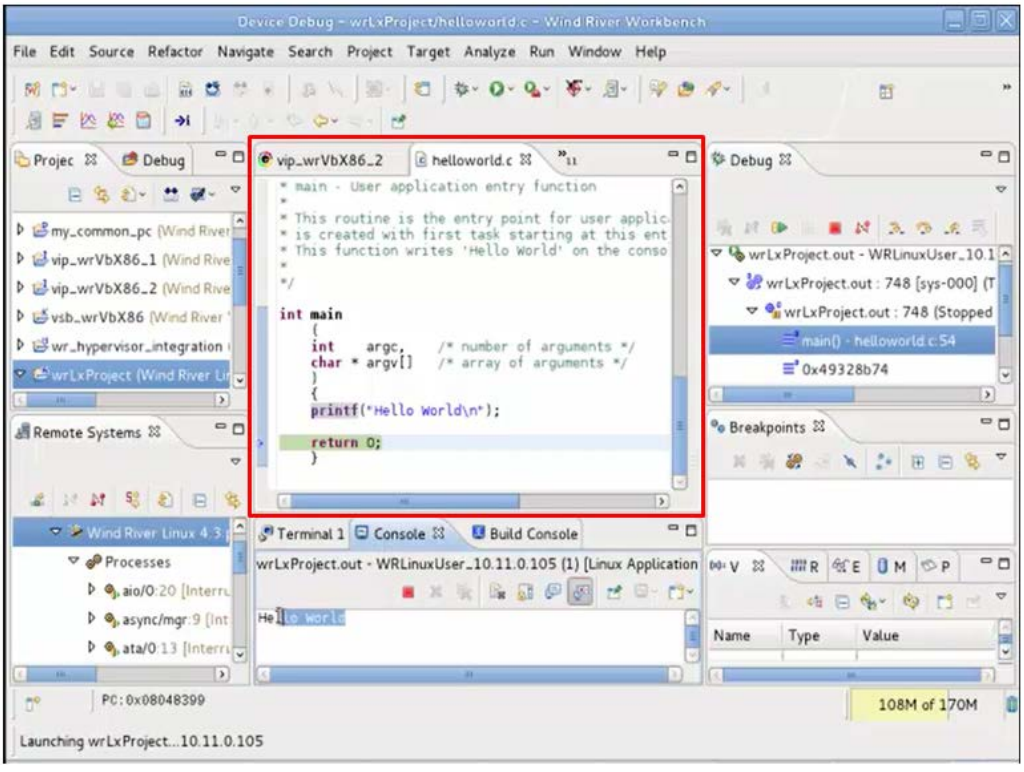
Any output the coding produces appears in the **Console** window at the bottom of Workbench. In this procedure, the **wrLxProject.out** application causes the words "Hello World" to print out. See Figure 136 – "Hello World".

Figure 136 – "Hello World"



The application or code console appears in the center of Workbench. See Figure 137 – Code Content.

Figure 137 – Code Content



9.6 Creating VxWorks Target Connection

Use these steps to enable the connection between workbench and a VxWorks image to allow the download and debug of an executable.

There are two possible ways to connect to a VxWorks GOS running on the target platform. If the vxWorks GOS has a physical Ethernet port configured, you can create a direct connection between Workbench and VxWorks using the NIC's IP address.

If on the other hand, the VxWorks GOS does not have a physical NIC assigned (or it is not configured), it can still connect to workbench via the VNIC interface. However, in this case you need to enable a proxy interface on the Linux GOS to forward the debug requests between the Linux interface and the VxWorks GOS. This is the option this workflow describes.

9.6.1 Setting Up Linux Proxy Agent

In order to begin a remote session with the VxWorks OS, Linux must be configured to forward messages to to it. Follow the procedure.

If not already, use Workbench to connect to the booted target platform.

Go to the Workbench Linux AMIO console.

At the Linux command prompt type

```
wrproxy &
```

Press **Enter**.

The proxy agent does not produce any output, but the proxy is active.

Next, the Linux user-mode agent described in section 1.2 must be activated if it is not already.

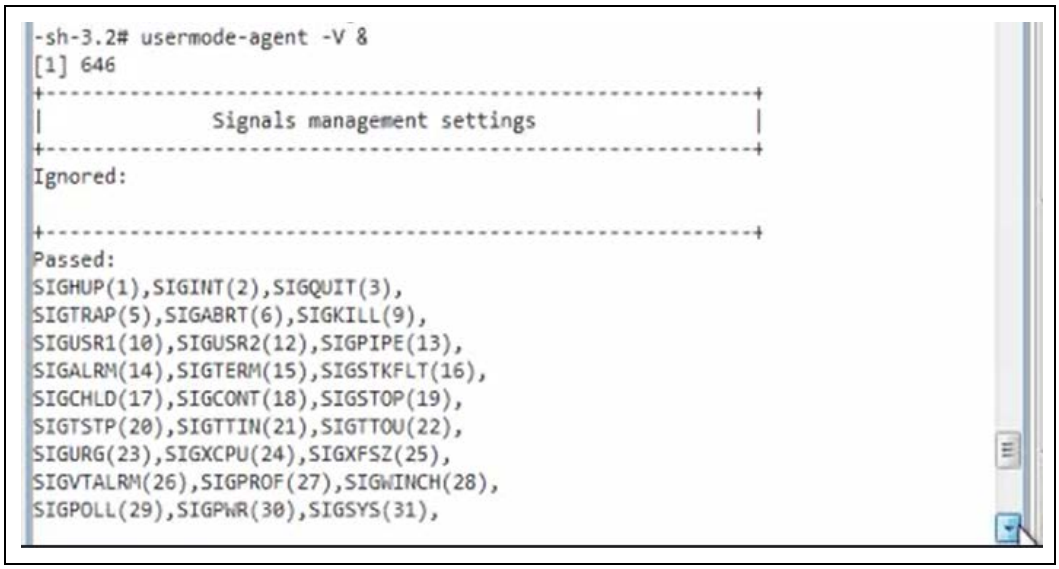
At the Linux command prompt type

```
usermode-agent – V &
```

Press **Enter**.

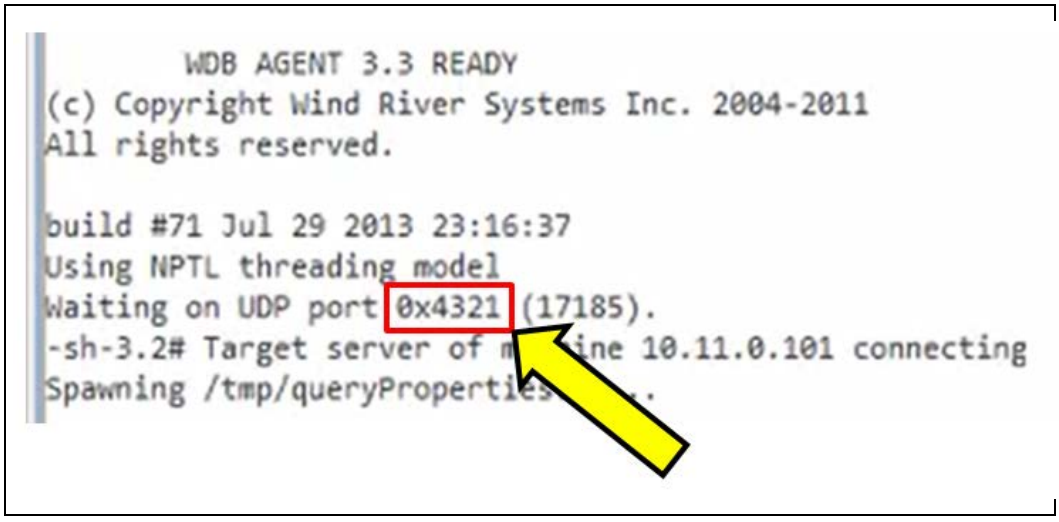
The console produces many lines of output. Figure 138 - Usermode Agent Output displays the top portion of this output.

Figure 138 - Usermode Agent Output



Scroll to the section of output that displays the UDP port number that the agent monitors. See the boxed item in Figure 139 - Agent Monitor Port.

Figure 139 - Agent Monitor Port

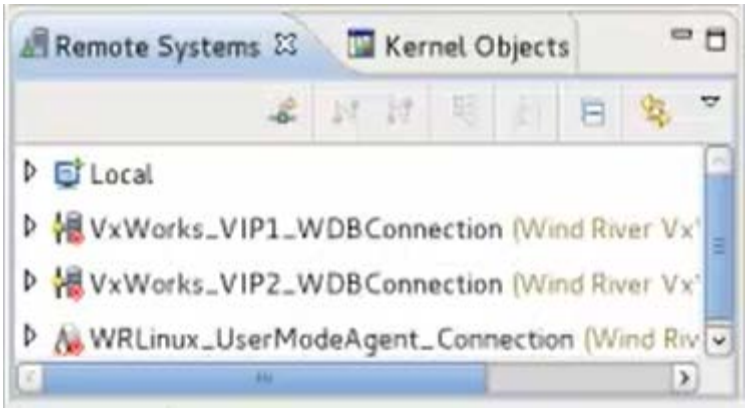

WDB AGENT 3.3 READY
(c) Copyright Wind River Systems Inc. 2004-2011
All rights reserved.
build #71 Jul 29 2013 23:16:37
Using NPTL threading model
Waiting on UDP port 0x4321 (17185).
-sh-3.2# Target server of
ine 10.11.0.101 connecting
Spawning /tmp/queryProperties

Record the port number.

9.6.2 Creating VxWorks Debug Connection

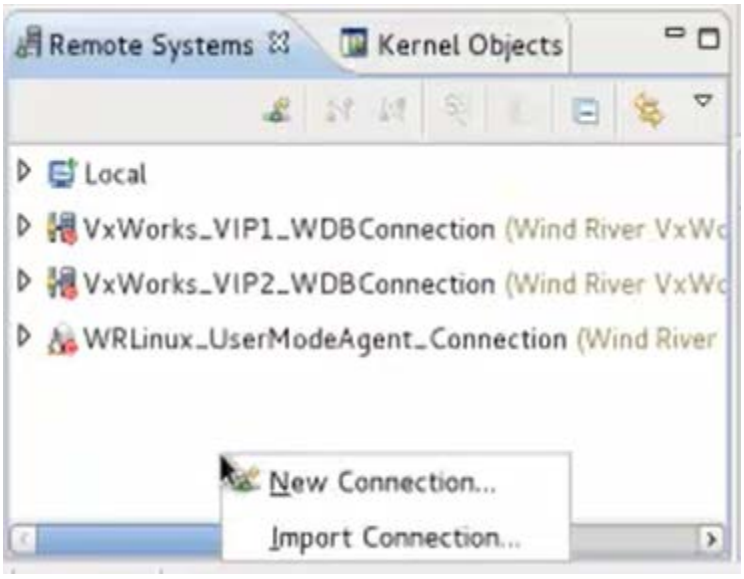
Go to the Workbench **Remote Systems** window. See Figure 140 – Remote Systems Window.

Figure 140 – Remote Systems Window



Right-click the white space within the **Remote Systems** window. A menu appears. See Figure 141 – White Space.

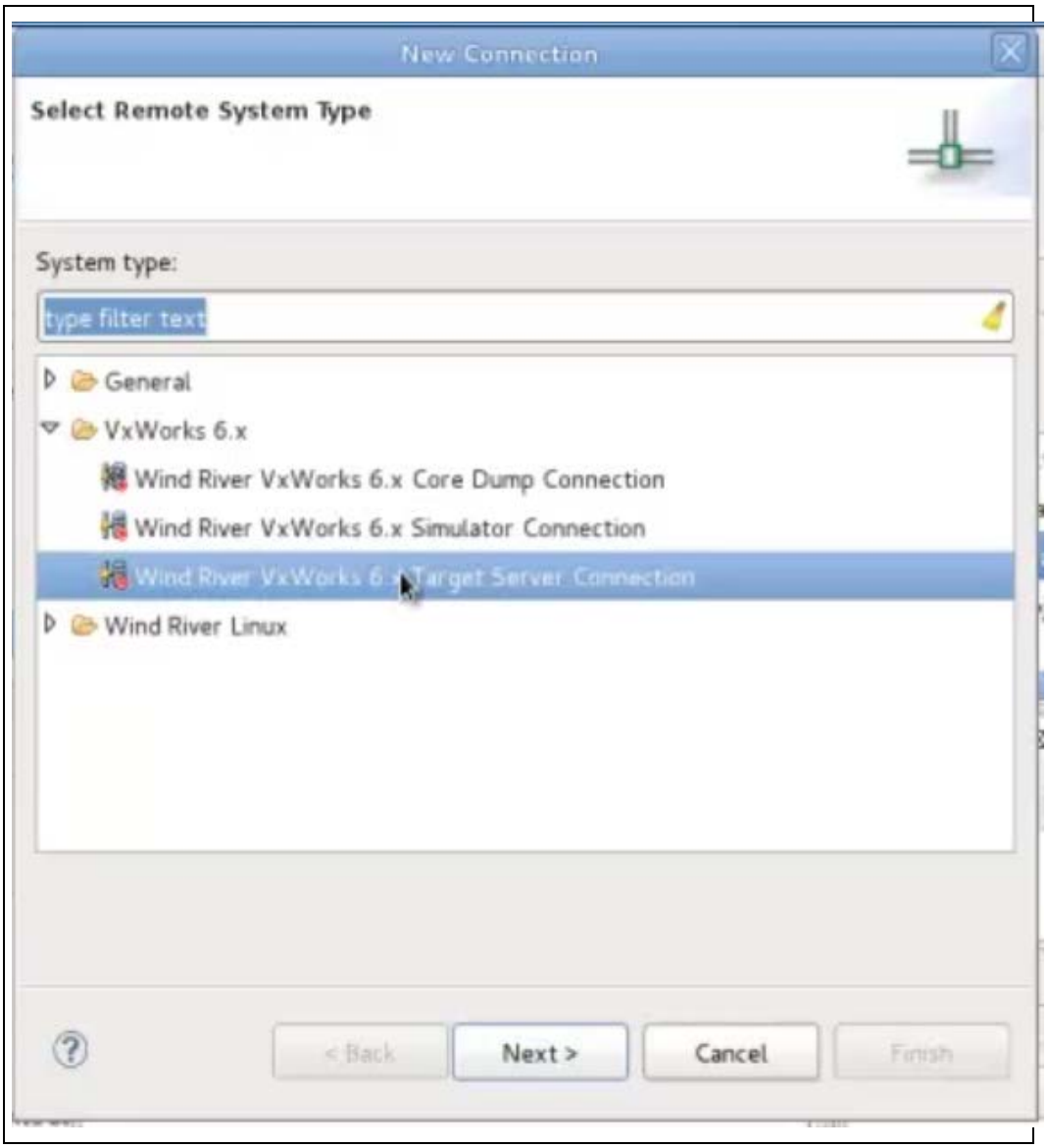
Figure 141 – White Space



Select **New Connection**. A **Select Remote System Type** window appears.

Select Wind River VxWorks 6.x Target Server Connection. See Figure 142 – System Type.

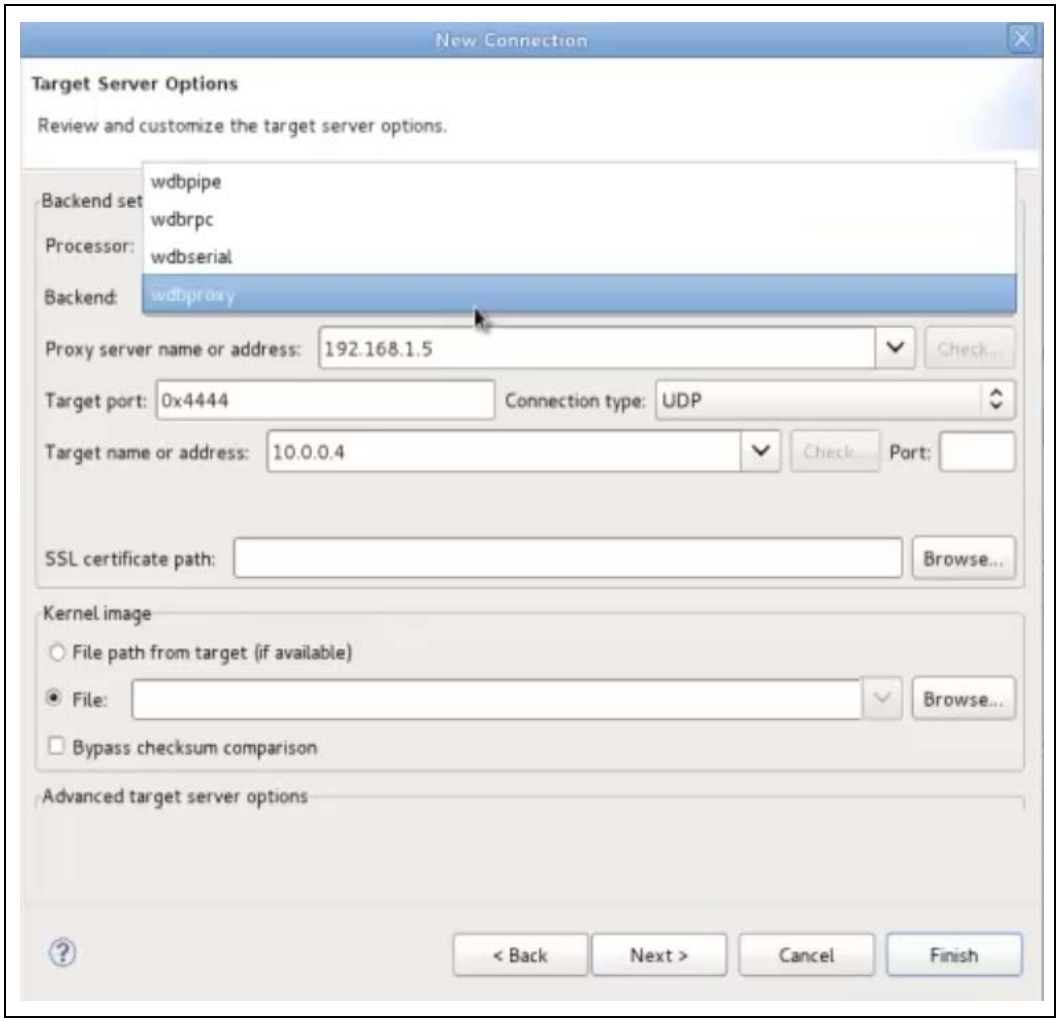
Figure 142 – System Type



Click **Next**. A **Target Server Connection for VxWorks** window appears.

wdbproxy should be selected for the **Backend** field. See Figure 143 – Backend.

Figure 143 – Backend



Into the **Proxy server name or address:** field, enter the **Linux OS IP** address. Figure 124 – Target Server…. uses **192.168.1.5** as the IP address.

NOTE: For more information about identifying or configuring the IP addresses for target platform OSes, see Section 10 – Workflow 5: Assign Persistent Target Platform OS IP Addresses.

Into the **Port:** field type the port number you recorded previously from the linux user-mode agent.

Into the **Target name or address:** field, type VNIC IP address of the VxWorks GOS you want to talk to.

At the **Kernel Image** section **File** field, browse to the VxWorks image used by the hypervisor corresponding to the Guest OS you want to connect to.

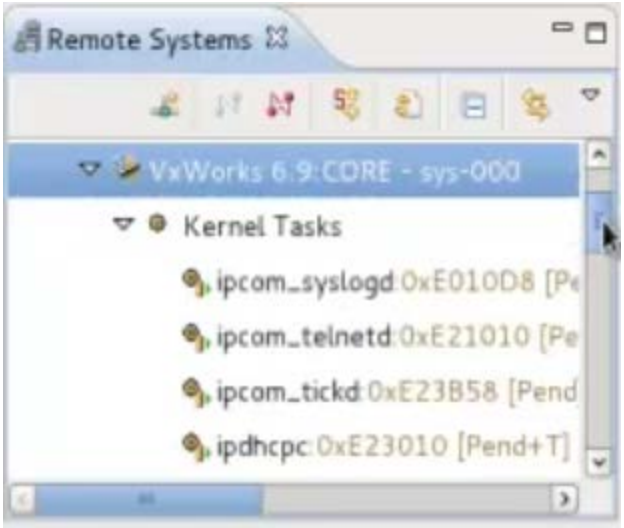
Click **Finish**. The development host connects to the target platform and a new connection icon appears in the **Remote Systems** window. A remote connection is established. See the arrowed item in Figure 144 – Remote Connection.

Figure 144 – Remote Connection

Expand the icons under the new connection until you see the **Kernel Tasks** object.

Optionally expand the **Kernel Tasks** icon to view the various OS tasks. See Figure 145 - VxWorks Live Tasks.

Figure 145 - VxWorks Live Tasks



Expand the icons under the new connection until you see the **Kernel Tasks** object.

9.7 Create a Sample VxWorks Application

Go to **File → New → Project**. A new Project window opens.

Select **Examples..**.

Click **Next**. From the **New Project Sample** wizard, select **The C demo demonstration Program**.

Click **Finish**. A **c_demo (Wind River VxWorks 6.9 Downloadable Kernel Module Project)** object appears in **Project Explorer**.

Right click **c_demo** project and select **Build Project**.

After a few moments, in the **Build Console** tab a message appears stating **Build Finished in 'c_demo'**

9.8 Download Application to a VxWorks Platform OS

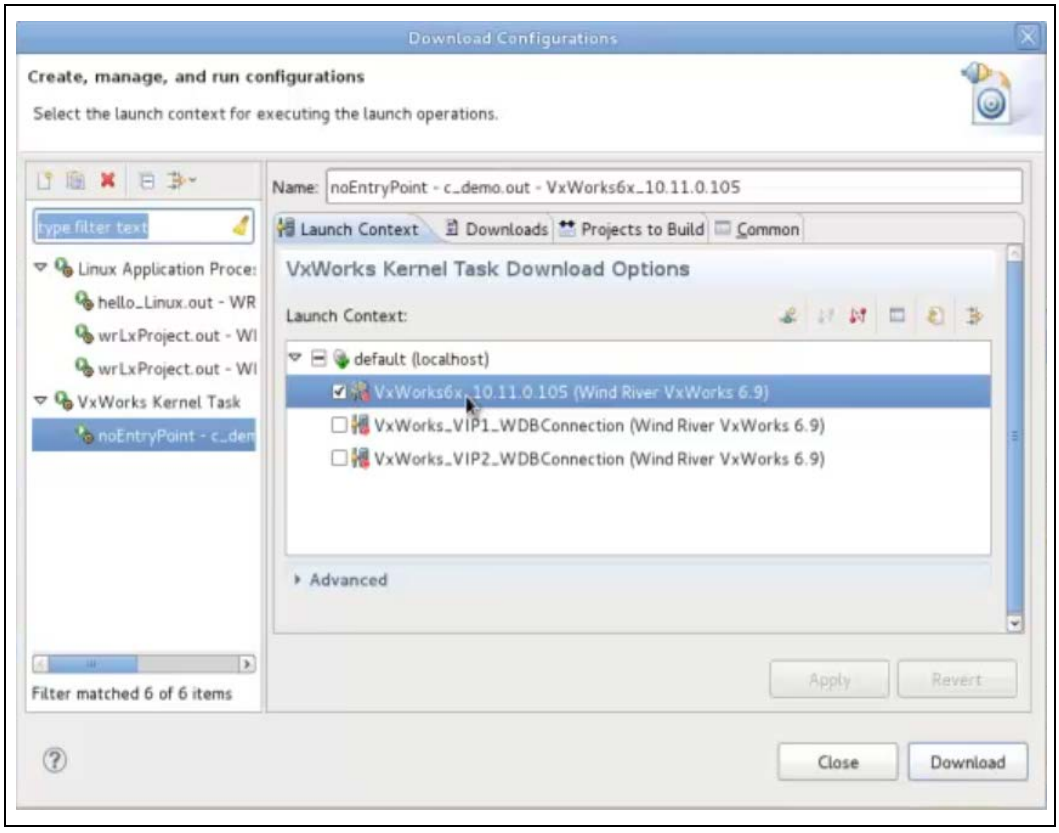
Follow this procedure to download code modules to the VxWorks platform.

NOTE: In this case "download" means a download of the compiled Downloadable Kernel Module to the target platform. Unlike the Linux image, this download is not persistent. On reset, the original image is restored. To make the download persistent, you would need to integrate the project to your VxWorks Image Project and recompile the hypervisor image.

In Workbench Project Explorer, right-click **c_project** and select **Download VxWorks Kernel Task**

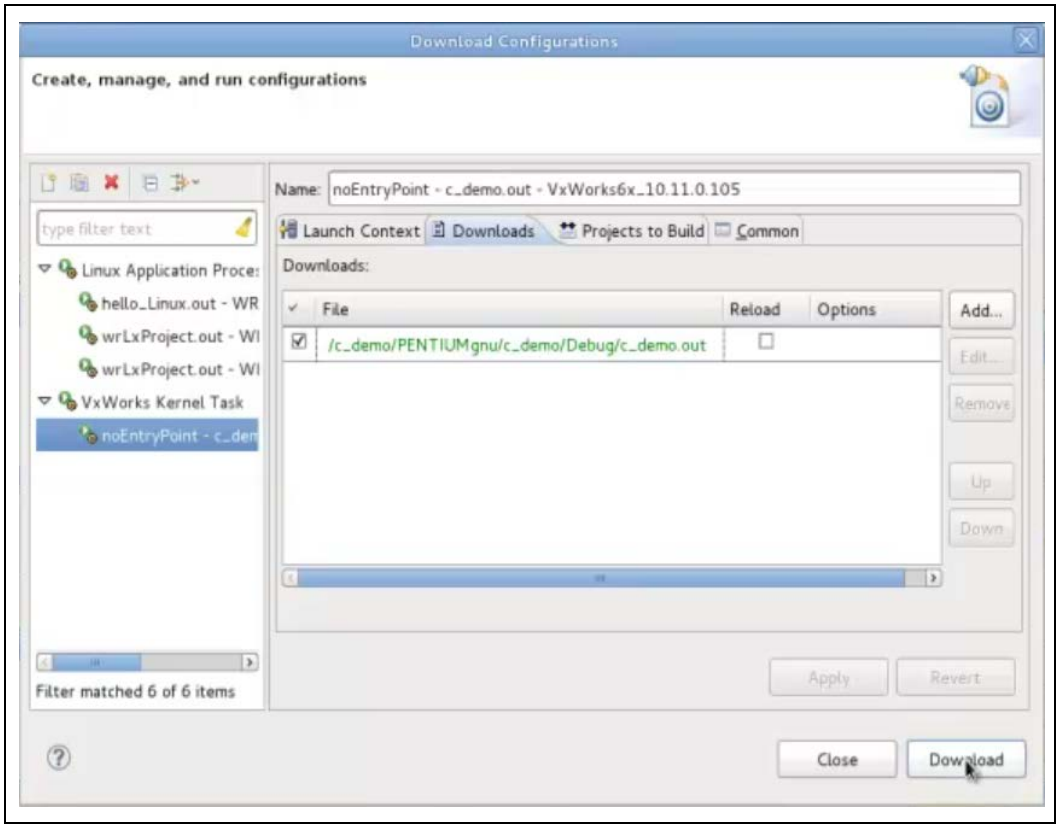
In the **Create, manage, and run configurations** window, in the **Launch Context:** tab, select the checkbox representing the vxWorks target platform connection. See the checked item in Figure 146 – Download Configuration.

Figure 146 – Download Configuration



Click the **Downloads** tab. See Figure 147 - Downloads Tab.

Figure 147 - Downloads Tab



Unlike Linux executables, a downloadable kernel module is akin to a code library that will be loaded in RAM. There is no path to download it to. The DKM is preconfigured since we are doing a download from the downloadable kernel module project. If you have another existing DKM that is not part of a workbench project, you could use the “Edit” button to replace the file, or even use “Add…” to download multiple DKMs.

This movement shows the right side of the **File Copy / Deploy** window and three buttons:

- **Add…** – Click this button to create additional DKM to download.
- **Edit…** – To use this function, select an existing download configuration and click the **Edit…** button. This allows you to change the DKM to download.
- **Remove** – To use this function, select an existing download configuration. Click the **Remove** button to permanently remove that download configuration.

This procedure assumes you will edit and run an existing line item.

Click **Download**.

The system downloads the contents from the development host to the target platform. As the content transfers from host to target, on the bottom right of Workbench a

console appears window showing the progress of the download and the content file structure build.

- If the download completes correctly, the console displays a **Build Finished** statement at the bottom of the console and a new line appears in the VxWorks connection, containing the DKM's filename. See Figure 148 – Successful DKM Download.
- If the download does not complete correctly, the console displays an error statement at the bottom of the console, with an indication of what the error entails.

Figure 148 – Successful DKM Download

Run the content and view the results.

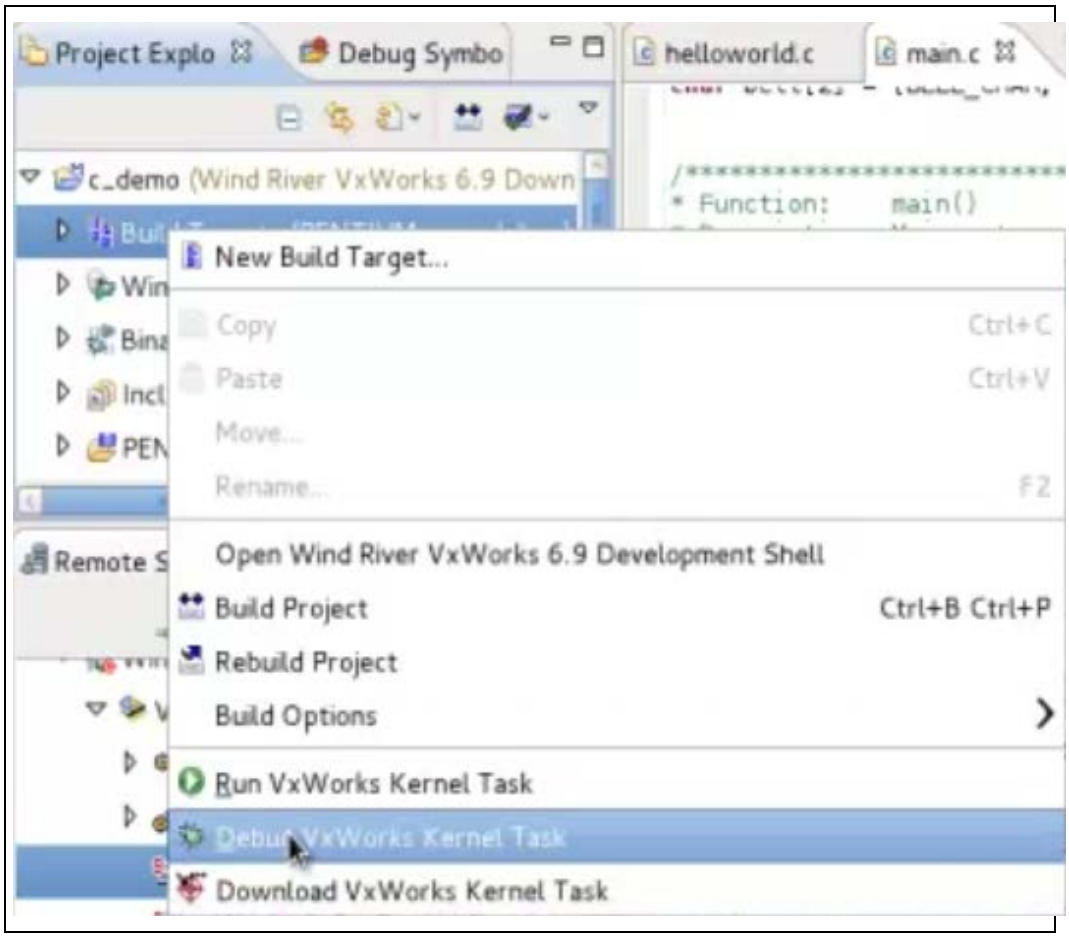
9.9 Debug VxWorks Target Platform Content

Use these steps to perform a debug procedure for a simple DKM. We use the project created in Section 9.8

In Workbench Project Explorer, select the **c_demo** project.

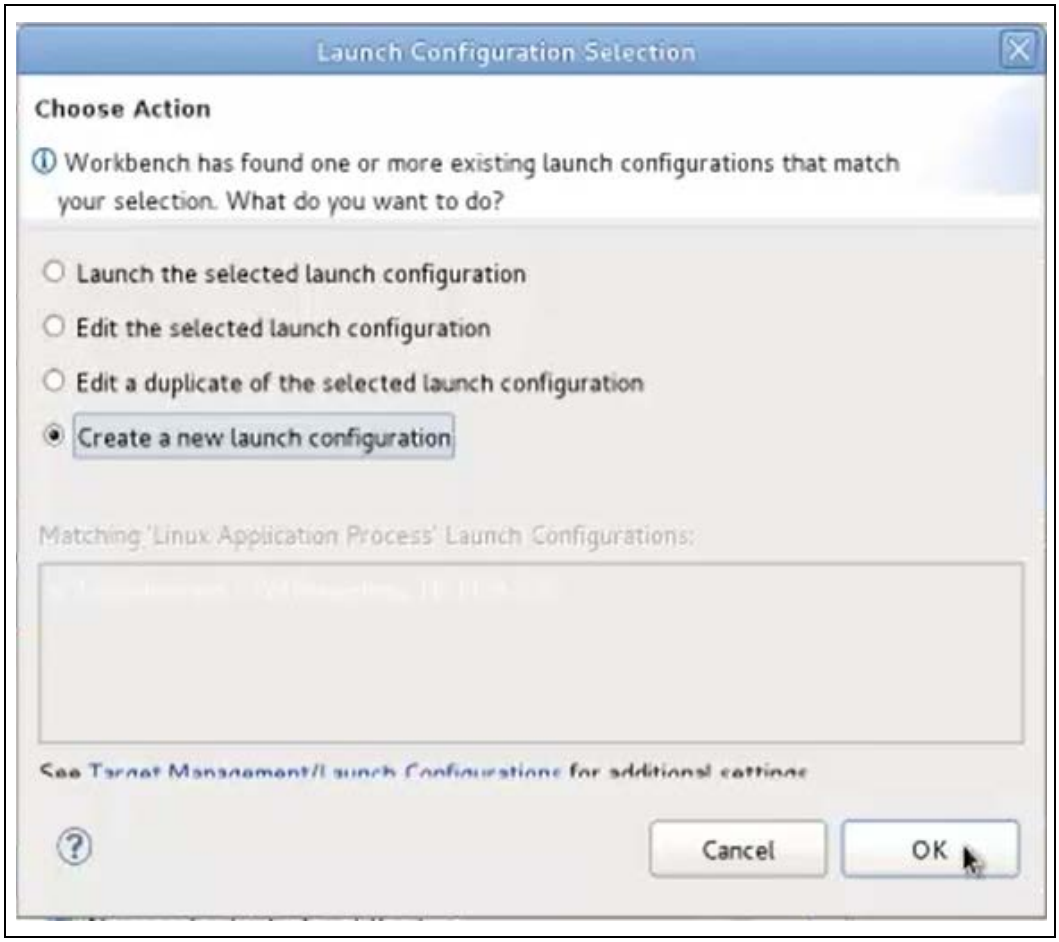
Right-click the Linux project and select **Debug VxWorks Kernel Task**. See Figure 149 – Debug Menu Item.

Figure 149 – Debug Menu Item



A **Choose Action** window appears. See Figure 150 – Choose Action Window.

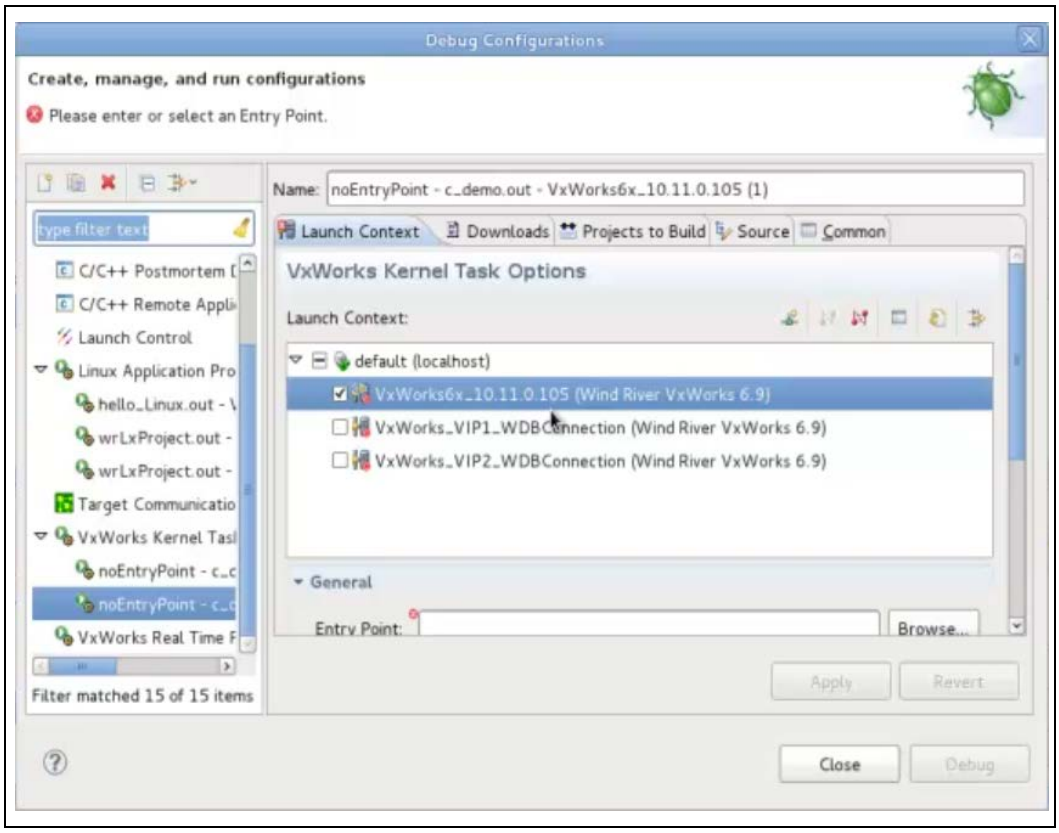
Figure 150 – Choose Action Window


Launch Configuration Selection
Choose Action
Workbench has found one or more existing launch configurations that match your selection. What do you want to do?
Launch the selected launch configuration
Edit the selected launch configuration
Edit a duplicate of the selected launch configuration
Create a new launch configuration
Matching 'Linux Application Process' Launch Configurations:
Cancel
OK

Select **Create a new launch configuration** and click **OK**. A Debug Configurations window appears. See Figure 151 – Debug Configurations.

NOTE: This window appears almost identical to the **Launch Context** window shown in Figure 127 – Edit File Copy/Deploy Context.

Figure 151 – Debug Configurations

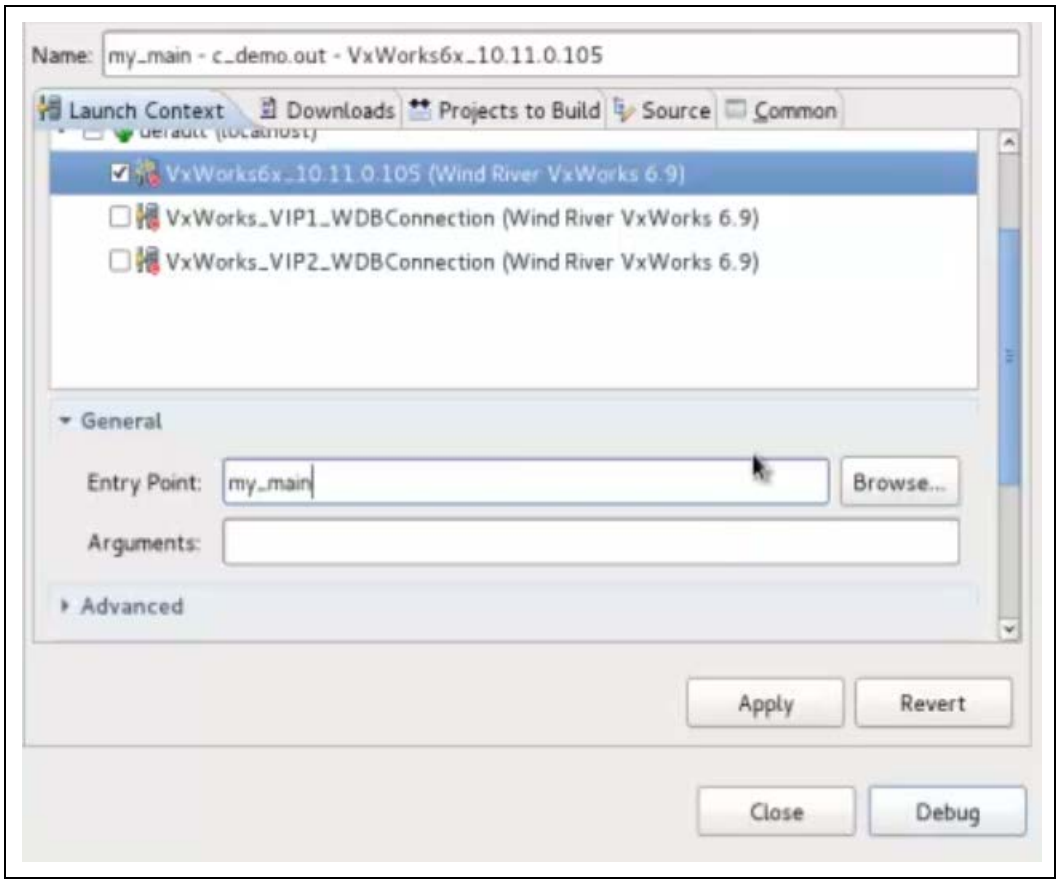


Select the target connection to use for debugging.

NOTE: unlike an executable, since a DKM is like a library, we must specify the name of the function we want to debug.

Into the Entry Point: field, enter the function you want to debug. If you are using the demo program, that would be my_main. See Figure 152 – Entry Point Selection.

Figure 152 – Entry Point Selection



Click **Debug**. In Workbench a **Debug** console appears at the right, and an output **Console** appears at the bottom.

At this point, you are ready to debug your code in exactly the same way and with the same features used in the Linux example in Section 9.5 – Debug Linux Target Platform Content.

§

10 Workflow 5: Assign Persistent Target Platform OS IP Addresses

10.1 About This Chapter

The Intel® Industrial Solutions System Consolidation Series development host uses *Wind River* Workbench* as its primary development tool. Use the Workbench development platform for innumerable development activities called *workflows*.

IMPORTANT: Before you perform this workflow, you must perform Workflow 1. Intel Corporation also recommends performing all other workflows previous to this workflow in the order they occur in this document.

This chapter includes both system *background information* and *instructions* for exploring system features.

- FIRST… Read and understand this chapter entirely.
- THEN… Re-read the chapter and perform the instructions in the workflow.

In this workflow, you configure the IP addresses for the VxWorks* virtual machine OSes on the target platform, and then confirm that these VMs can communicate within the system's virtual network (VNIC).

10.2 Default OS IP Address assignment

The Intel® Industrial Solutions System Consolidation Series (SCS) four virtual platforms are networked together via a virtual Layer 2 switch. Each individual virtual platform has a *virtual Network Interface Card* (*VNIC* in this document). Intel® Industrial Solutions System Consolidation Series (SCS) assigns target platform IP addresses to the four guest operating systems by default as follows:

- **Linux* –** 10.0.0.3
- **VxWorks 1 –** 10.0.0.4
- **VxWorks 2 –** 10.0.0.5

NOTE: Intel® Industrial Solutions System Consolidation Series currently supports four physical Ethernet interfaces. By default, each guest OS is assigned at least one physical Ethernet interface in addition to the VNIC.

10.2.1 Customizable Target OS IP Address Assignment

You can modify the guest OS address manually from a guest OS' command shell (accessible from the AMIO terminal) by using the ifconfig command. However, such a change is not persistent. A target platform reboot restores the default IP addresses (10.0.0.3 through 10.0.0.5). You can permanently override the guest OS boot IP addresses by configuring them using the development host Workbench software.

10.2.2 Customizable Target OS IP Address Assignment

10.2.2.1 Configure a Persistent VxWorks IP Address – 1st Instance

When booting the target platform, you may manually make a change to a VxWorks OS using the **ifconfig** at the command prompt. However, these changes are not persistent and will be reset to the default IP address the next time the target platform reboots.

To enable a persistent IP address for VxWorks **vip_wrVbx86_1**, follow the procedure below.

Go to the **vip_wrVbx86_1** icon and expand it.

Go to the **usrAppInit.c** object.

Double-click **usrAppInit.c**. A **usrAppInit.c** work pane opens.

Double-click the **usrAppInit.c** tab to expand its work pane.

Under the **/* add application specific code here */** line, type:

```
ipAttach(0, “gei”);
ifconfig(“gei0 10.11.0.1 netmask 255.255.255.0 up”);
```

where **10.11.0.1** is an IP address not already used by another OS. Modify the IP address as needed. See Figure 153 – usrAppInit.c IP Change 1.

Figure 153 – usrAppInit.c IP Change 1

```
37 #ifdef  USER_APPL_INIT
38     USER_APPL_INIT;     /* for backwards compatibility */
39 #endif
40
41     /* add application specific code here */
42     ipAttach(0,"gei");
43     ifconfig("rtg0 10.11.0.1 netmas 255.255.255.0 up");
44
45     }
46
47 #ifdef INCLUDE_USER_PRE_KERNEL_APPL_INIT
```

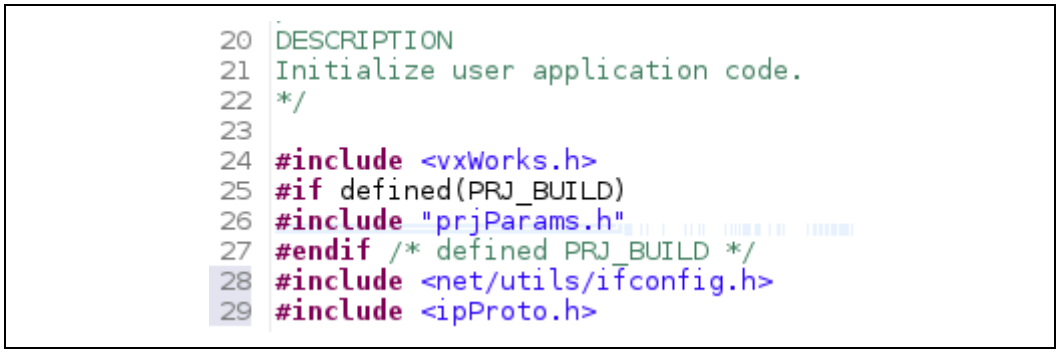
Locate the **/* DESCRIPTION Initialize user application code */** line.

Under this line is an existing entry that looks like the code below:

```
#include <VxWorks.h>
#if defined (PRJ_BUILD)
#include “prjParams.h”
#endif /* defined PRJ_BUILD */
```

Enter the extra two lines shown below this code entry so that the resulting code block appears as shown in Figure 154 - usrAppInit.c IP Change 2.

Figure 154 - usrAppInit.c IP Change 2

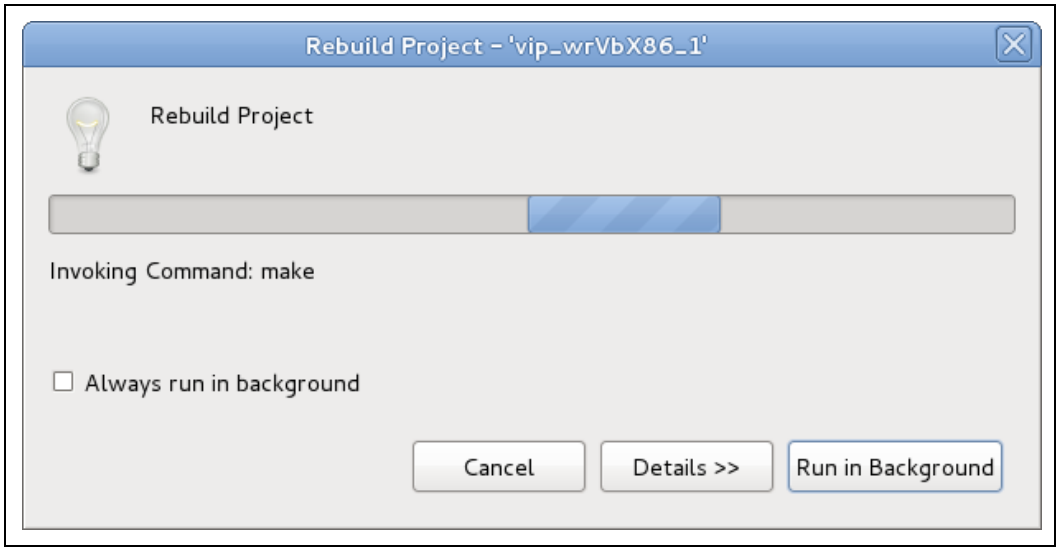


Press **CRTL+S** to save the VxWorks project file.

Right-click the **vip_wrVbx86_1** object and select **Rebuild Project**. The system rebuilds the project.

A **Rebuild Project** window appears as the system rebuilds the project. See Figure 155 - Rebuild VxWorks Project.

Figure 155 - Rebuild VxWorks Project



When the systems completes rebuilding, a message at the bottom of the build console states “Build finished in…”.

10.2.2.2 Configure a Persistent VxWorks IP Address – 2nd Instance

To enable a persistent IP address for VxWorks **vip_wrVbx86_2**, you essentially repeat the same procedure as before for the 2nd VxWorks image. The only difference is that the Ethernet interface is called **gei**, not rtg.

Go to the **vip_wrVbx86_2** icon and expand it.

Go to the **usrAppInit.c** object.

Double-click **usrAppInit.c**. A **usrAppInit.c** work pane opens.

Double-click the **usrAppInit.c** tab to expand its work pane.

Under the **/* add application specific code here */** line, type:

```
ipAttach(0, “rtg”);
ifconfig(“rtg0 10.11.0.2 netmask 255.255.255.0 up”);
```

where **10.11.0.2** is an IP address not already used by another OS. Modify the IP address as needed.

At this point, you can follow the same steps as in Section 10.2.2.1 – Configure a Persistent VxWorks IP Address – 1st Instance.

10.2.3 Rebuild Hypervisor and Boot Target Platform

Right-click the **wr_hypervisor_integration** object and select **Rebuild Project**.

The system gathers the changes you made to the VxWorks project, rebuilds the Hypervisor project, and writes an updated target platform configuration to a new **system.elf** file. When the systems completes rebuilding, a message at the bottom of the build console states “Build finished in…”.

10.2.4 Boot the Target Platform with Persistent IP Address Changes

CAUTION: Intel Corporation provides you with the target platform media USB for copying the system.elf file from the development host to the target platform. However, copying data onto the target platform USB media may cause the data on the media to be overwritten. Make certain to back up data before using the media. Failure to do so may result in the loss of the data on the media.

At the development host, insert a Linux-formatted USB media (with 16 GB or more free space) into an open USB port. The USB media mounts to the Linux file system.

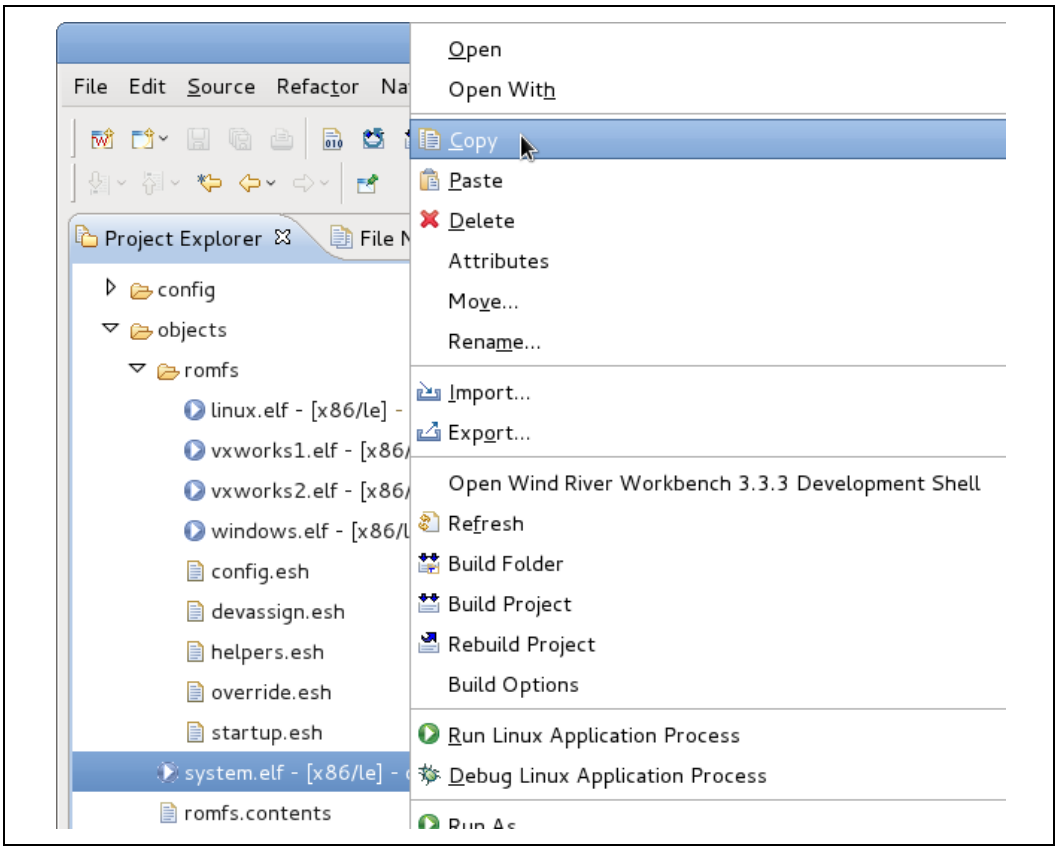
In Workbench Project Explorer, open the **wr_hypervisor_integration** object.

Find the **Binaries** folder and expand it.

In the **Binaries** folder, find and right-click the **system.elf** object.

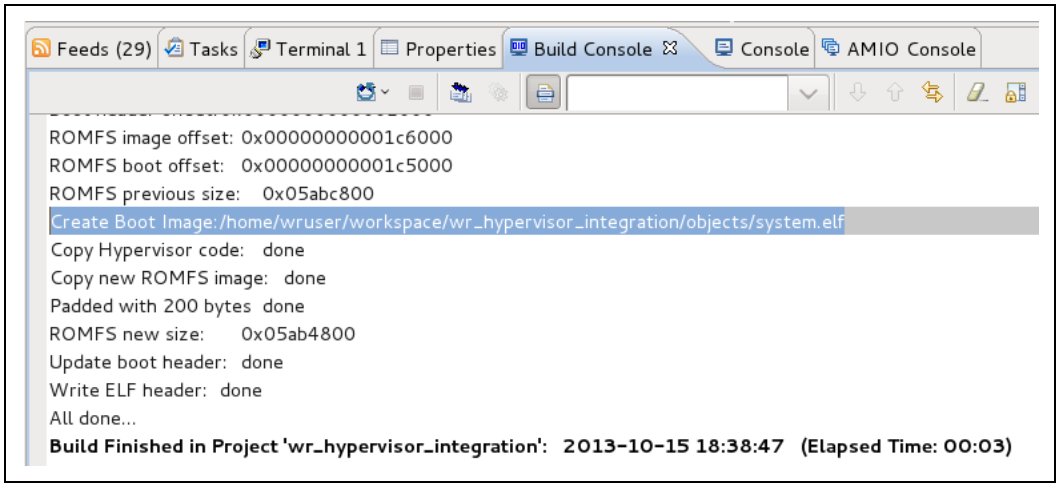
Select **Copy**. The **system.elf** file copies to the system. See Figure 156 – System.elf Copy.

Figure 156 – System.elf Copy



You may optionally use a command line or the Linux file explorer to copy the **system.elf** file. For the system file path, see the darkened content in Figure 157 – System.elf Directory.

Figure 157 – System.elf Directory



Minimize Workbench.

Use Linux File Explorer to navigate to the root directory of the USB media.

Paste the **system.elf** file to the root directory of the USB media.

After the file finishes copying, unmount the USB media and remove it from the development host.

Make certain that the target platform is powered down.

Ensure that no other bootable media are inserted into the target platform.

Insert the USB media into an open port on the target platform.

NOTE: For a faster boot, Intel Corporation recommends you connect the boot media to a USB 3.0 port on the target platform. Arrows in Figure 158 - USB 3.0 Ports provide the locations of two target platform USB 3.0 ports.

Figure 158 - USB 3.0 Ports

10.3 Configure Target Platform BIOS to Boot from USB Media

Before booting the target platform from USB media, you first configure the target platform BIOS to boot from USB media.

NOTE: To ensure that you start the target platform from USB media, follow this procedure each time you need to boot from USB media.

Shut down the target platform and wait 5 seconds.

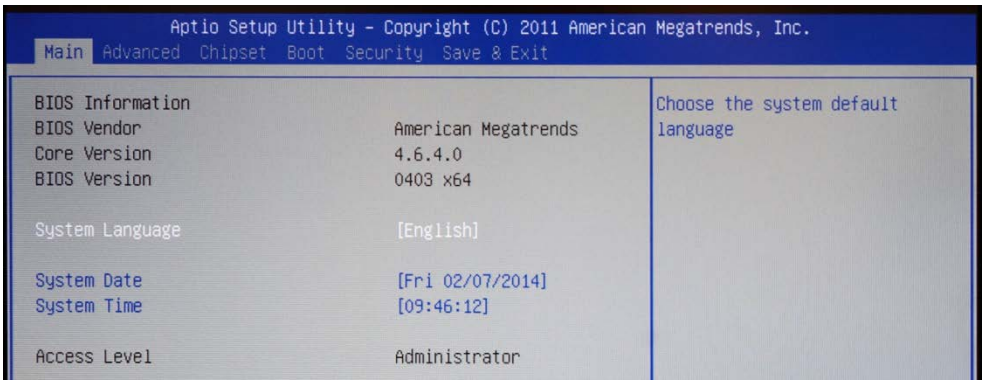
Place the bootable USB media into an open USB port (preferably a blue USB 3.0 port).

Press and release the target platform power button. The target platform begins to start.

As the target platform begins to start, repeatedly press the **DELETE** button.

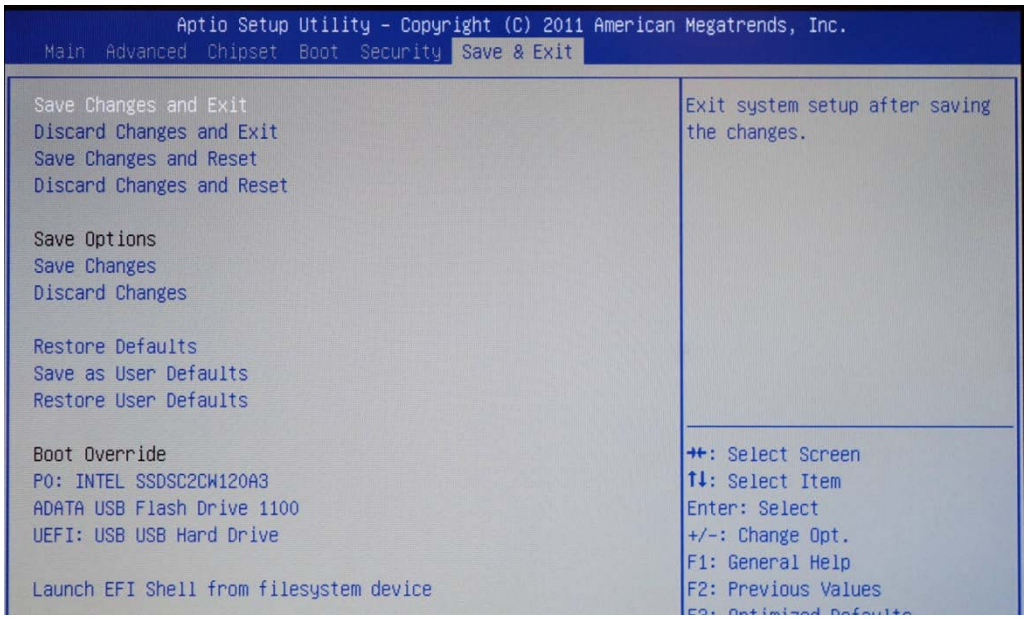
The target platform enters the BIOS **Main** tab. See Figure 159 – BIOS 1.

Figure 159 – BIOS 1



Use the right-arrow key to go to the **Save & Exit** tab. See Figure 160 – BIOS 2.

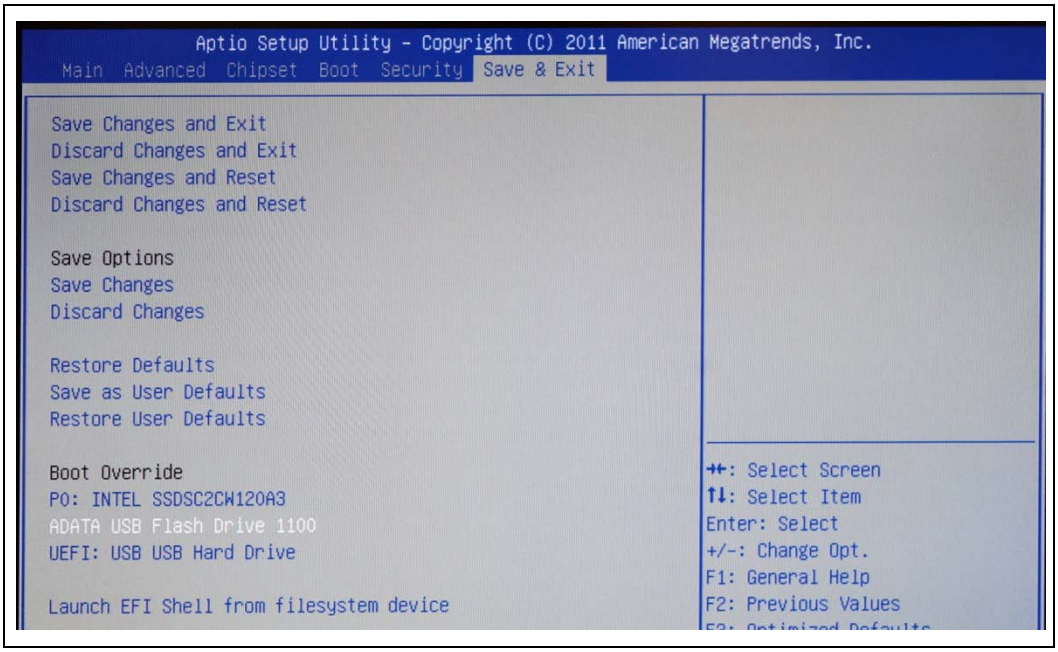
Figure 160 – BIOS 2



Use the down-arrow key to move the highlighted selection to the USB media. See Figure 161 – BIOS 3.

NOTE: In this example, the USB media is named **ADATA USB Flash Drive 1100**. Your USB media may have a different name.

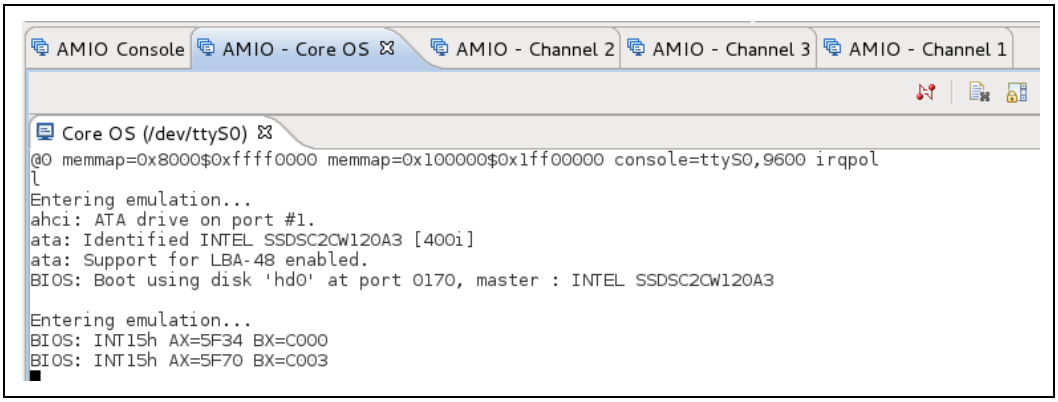
Figure 161 – BIOS 3



Press **Enter**. The target platform boots from USB media.

On the development host, connections to the guest OSes appear in individual AMIO consoles at the bottom of the development host Workbench workspace. See Figure 162 –AMIO Consoles.

Figure 162 –AMIO Consoles


AMIO Console
AMIO - Core OS
AMIO - Channel 2
AMIO - Channel 3
AMIO - Channel 1
Core OS (/dev/ttyS0)
@0 memmap=0x8000$0xffff0000 memmap=0x100000$0x1ff00000 console=ttyS0,9600 irqpol
l
Entering emulation...
ahci: ATA drive on port #1.
ata: Identified INTEL SSDSC2CW120A3 [400i]
ata: Support for LBA-48 enabled.
BIOS: Boot using disk 'hd0' at port 0170, master : INTEL SSDSC2CW120A3
Entering emulation...
BIOS: INT15h AX=5F34 BX=C000
BIOS: INT15h AX=5F70 BX=C003

In the VxWorks 1 AMIO console type **ifconfig** and press **Enter**.

In the output that appears, confirm that the IP address you entered is assigned to the OS.

Repeat this procedure for other OSes with IP address changes.

NOTE: For Linux, type **ifconfig –a**.

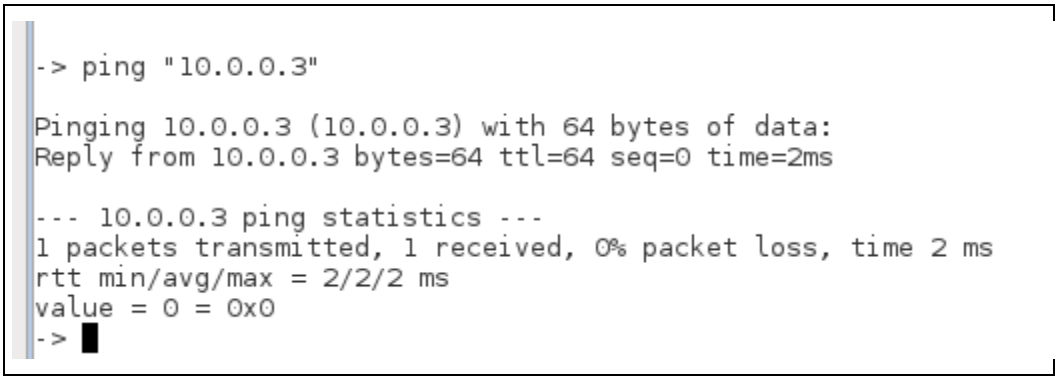
10.3.1 VxWorks-to-Linux Communication

To demonstrate that a communication path exists from VxWorks to Linux, visible from the VxWorks side, use the ping command.

At either VxWorks console, double-click the console tab. The console expands.

At the command line, type **ping "10.0.0.3"** and press **Enter**. See Figure 163 - Ping Command.

Figure 163 - Ping Command


-> ping "10.0.0.3"
Pinging 10.0.0.3 (10.0.0.3) with 64 bytes of data:
Reply from 10.0.0.3 bytes=64 ttl=64 seq=0 time=2ms
--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2 ms
rtt min/avg/max = 2/2/2 ms
value = 0 = 0x0
->

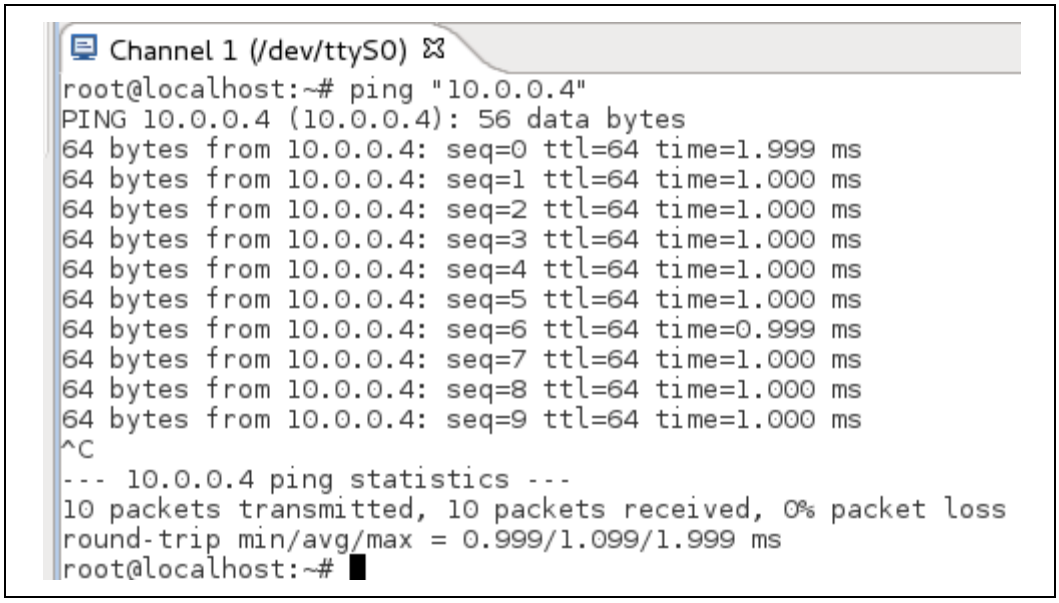
A response from Linux appears indicating that a reply was received.

10.3.2 Linux-to-VxWorks Communication

In the AMIO console at the Linux command line, type **ping "10.0.0.4"** and press **Enter**.

A response from Linux appears indicating that a reply was received. See Figure 164 - Ping Command.

Figure 164 - Ping Command


Channel 1 (/dev/ttyS0)
root@localhost:~# ping "10.0.0.4"
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: seq=0 ttl=64 time=1.999 ms
64 bytes from 10.0.0.4: seq=1 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=2 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=3 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=4 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=5 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=6 ttl=64 time=0.999 ms
64 bytes from 10.0.0.4: seq=7 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=8 ttl=64 time=1.000 ms
64 bytes from 10.0.0.4: seq=9 ttl=64 time=1.000 ms
^C
--- 10.0.0.4 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.999/1.099/1.999 ms
root@localhost:~#

Press **CRTL+C** to stop the ping process.

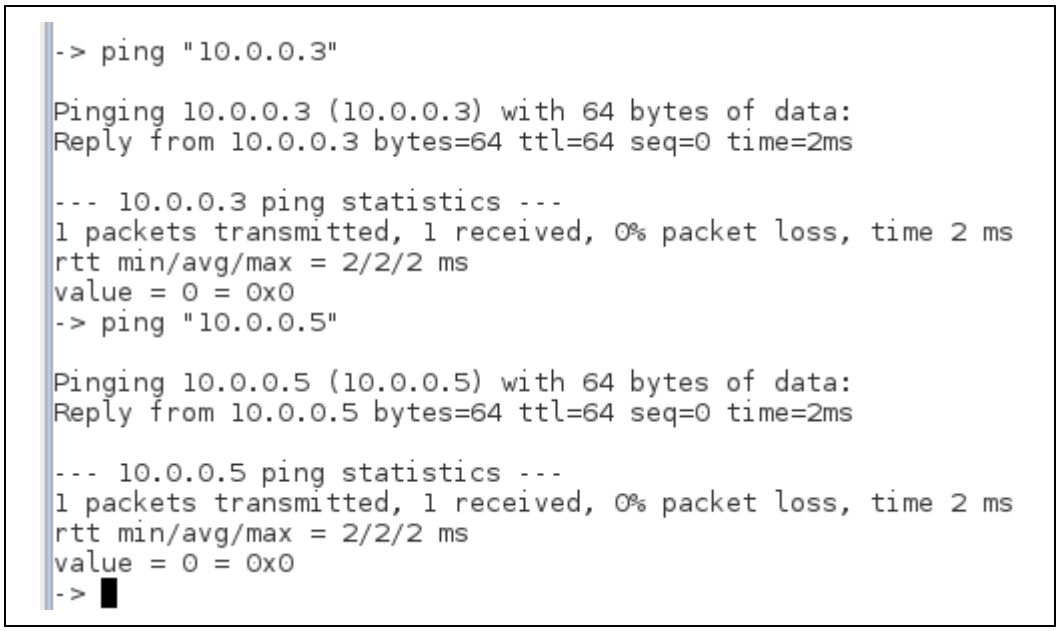
10.3.3 VxWorks-to-VxWorks Communication

To demonstrate that a communication path exists between the two VxWorks instances, use the ping command.

At the VxWorks 1 command line, type **ping "10.0.0.5"** and press **Enter.**

A response from VxWorks 2 indicates that a reply was received. See Figure 165 - Ping Command.

Figure 165 - Ping Command


-> ping "10.0.0.3"
Pinging 10.0.0.3 (10.0.0.3) with 64 bytes of data:
Reply from 10.0.0.3 bytes=64 ttl=64 seq=0 time=2ms
--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2 ms
rtt min/avg/max = 2/2/2 ms
value = 0 = 0x0
-> ping "10.0.0.5"
Pinging 10.0.0.5 (10.0.0.5) with 64 bytes of data:
Reply from 10.0.0.5 bytes=64 ttl=64 seq=0 time=2ms
--- 10.0.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2 ms
rtt min/avg/max = 2/2/2 ms
value = 0 = 0x0
->

A response from VxWorks appears indicating that a reply was received.

Repeat the procedure from the other VxWorks command prompt.

§

11 Workflow 6: Configure a New Device

11.1 About This Chapter

The Intel® Industrial Solutions System Consolidation Series development host uses *Wind River Workbench* as its primary development tool. Use the Workbench development platform for innumerable development activities called *workflows*.

IMPORTANT: Before you perform this workflow, you must perform Workflow 1. Intel Corporation also recommends performing all other workflows previous to this workflow in the order they occur in this document.

This chapter includes both system *background information* and *instructions* for exploring system features.

- FIRST… Read and understand this chapter entirely.
- THEN… Re-read the chapter and perform the instructions in the workflow.

In this workflow, you have the opportunity to learn how to assign devices to specific Guest OSes using Hypervisor to modify a script file that controls devices. Afterwards, you have the opportunity to add and assign a mini PCIe device.

11.1.1 Special Version of Hypervisor

Engineers who have worked previously with hypervisor technology know that out-of-box systems that have not been built for a specific hardware platform require much time for adaptation.

To address this problem, Intel® Industrial Solutions System Consolidation Series (SCS) comes with a special implementation of Hypervisor that is optimized for and runs on the system's target hardware platform. The SCS Hypervisor is somewhat static in the sense that it allocates certain resources (such as RAM to the VMs) in order to provide stability and dependability for high-volume production environments. These allocations are static: You should not change their configuration, lest you risk degraded performance or inoperability.

While the SCS system provides the stability through the unmodifiable portions of its configuration, its Hypervisor layer is also somewhat flexible to your needs. Included in this flexibility is the ability to discover and assign certain devices to specific OSes.

11.1.2 Hypervisor Resource Assignment

When Hypervisor starts up on the target platform, one of many operations it performs is device discovery. This entails:

- discovering essential system resources (such as RAM) and non-essential physical devices (such as USB controllers).
- mapping each resource to a specific type and class.
- mapping physical devices and resources to specific OSes.
- mapping all unassigned devices to a default OS.

During this routine, Hypervisor classifies devices as shown in Table 5 - Device Classifications.

Table 5 - Device Classifications

Resource Type	Signifier	PCI Device Class	PCI Device Subclass
Serial	(ser)	Class 7	Sub 0
Ethernet	(eth)	Class 2	Sub 0
VGA	(vga)	Class 3	Sub 0
USB Controller¥	(usb)	Class 12	Sub 3
IDE	(ide)	Class 1	Sub 1
RAID	(raid)	Class 1	Sub 4
SATA	(sata)	Class 1	Sub 6

¥ Hypervisor classifies USB controllers only, not individual USB devices and sub-connections.

Because there may be more than one of the same device type, Hypervisor further identifies a device by appending a number sequentially (starting with zero) after its signifier. For example:

- ser0, ser1, ser2…
- ide0, ide1, ide2…

The system processes this information and assigns two kinds of resources:

- **Static resources** – These are devices and resources you cannot configure, such as RAM, COM1 & COM2 serial ports and the number of Guest OSes.
- **User-configurable resources** – These are devices you can configure, such as USB controllers and Ethernet. These resource configurations reside in scripts that you can modify. Generally speaking, any PCI device discovered on a bus is user-configurable.

Read on to learn about user-configurable resources.

11.1.3 User-Configurable Resources

Use Workbench development software to modify OS access to certain *types* of devices, as well as *specific* devices. Table 6 - User- Configurable Resources lists all of the devices that come with Intel® Industrial Solutions System Consolidation Series target hardware.

Table 6 - User- Configurable Resources

Resource Type	Resource "Signifier"	Bus/Dev/Fn	VendId:DevId
Host Bridge	-	B0/D00/F0	0x8086:0x0104
VGA Display	vga0	B0/D02/F0	0x8086:0x0116
SCC Com. Ctlr	-	B0/D22/F0	0x8086:0x1c3a
Serial Com. Ctlr	*	B0/D22/F3	0x8086:0x1c3b
Ethernet Controller	eth0	B0/D25/F0	0x8086:0x1502
USB Serial Bus	usb0	B0/D26/F0	0x8086:0x1c2d
Multi-Media Device	-	B0/D27/F0	0x8086:0x1c20
PCI Bridge	-	B0/D28/F0	0x8086:0x1c10
PCI Bridge	-	B0/D28/F4	0x8086:0x1c18
PCI Bridge	-	B0/D28/F5	0x8086:0x1c1a
PCI Bridge	-	B0/D28/F6	0x8086:0x1c1c
PCI Bridge	-	B0/D28/F7	0x8086:0x1c1e
USB Serial Bus	usb1	B0/D29/F0	0x8086:0x1c26
ISA Bridge Device	-	B0/D31/F0	0x8086:0x1c4f
SATA Controller	Sata	B0/D31/F2	0x8086:0x1c03
SMBus Serial Bus	-	B0/D31/F3	0x8086:0x1c22
Ethernet Controller	eth1	B2/D00/F0	0x8086:0x10d3
Ethernet Controller	eth2	B3/D00/F0	0x10ec:0x8168
Ethernet Controller	eth3	B4/D00/F0	0x10ec:0x8168
USB Serial Bus	usb2	B5/D00/F0	0x104c:0x8241

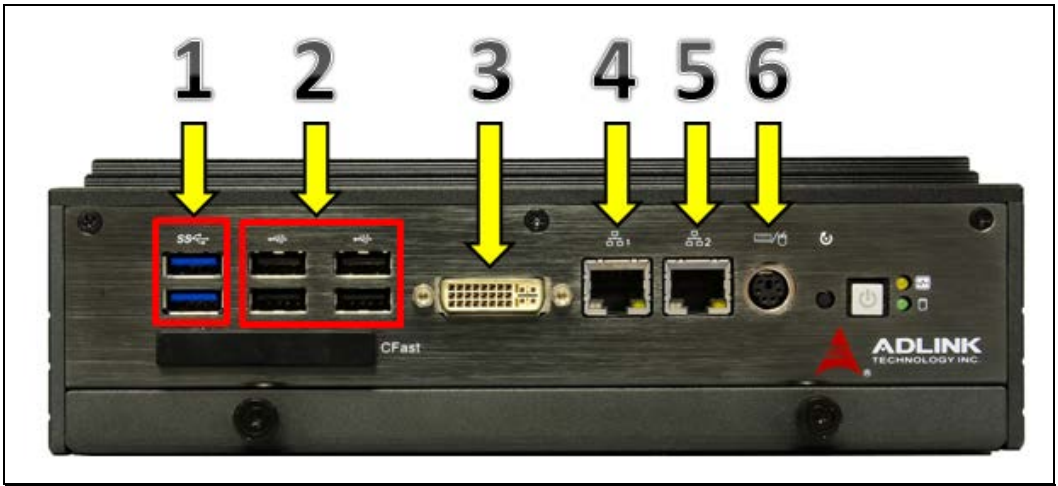
NOTE: On the ISSCS host, there is a single serial controller. Because it is not possible to share a single device across multiple guest OSes, the serial ports are statically

assigned and the systems uses AMIO to multiplex serial access to the multiple guest OSes.

11.1.4 Default Physical Device Allocations

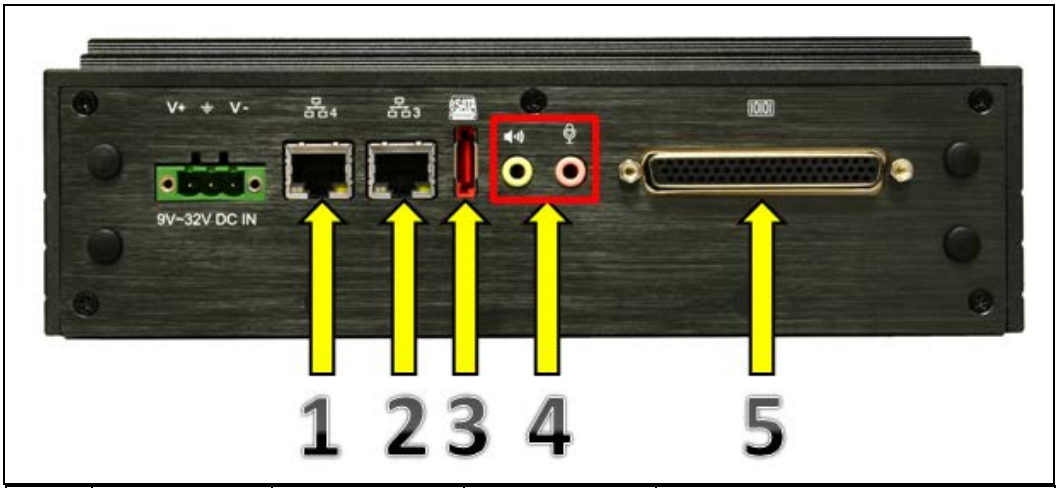
Intel® Industrial Solutions System Consolidation Series SCS comes with preconfigured physical device allocations as shown in Figure 166 – Target Platform Devices 1 and Figure 167 - Target Platform Devices 2.

Figure 166 – Target Platform Devices 1



Item #	Device Type	Assigned to VM	ifconfig ID	Comment
1	USB 3.0	Linux	(NA)	Denoted by blue connector color.
2	USB 2.0	VxWorks 1	(NA)	Denoted by black connector color.
3	Video	Linux	(NA)	Basic (non-graphical) command-line style video only supported at this time.
4	Ethernet	VxWorks 1	gei0	“gei” indicates this device is driven by a Intel driver.
5	Ethernet	Linux	eth0	“gei” indicates an Intel Corporation driver.
6	PS2	(NA)	(NA)	Not supported by this implementation of SCS.

Figure 167 - Target Platform Devices 2



Item #	Device Type	Assigned to VM	ifconfig ID	Comment
1	Ethernet	VxWorks 2	rtg	"rtg" indicates this device is driven by a Realtek driver.
2	Ethernet	Linux	eth1	"rtg" indicates this device is driven by a Realtek driver.
3	eSATA	(NA)	(NA)	Not supported by this implementation of SCS.
4	Audio	(NA)	(NA)	Not supported by this implementation of SCS.
5	Serial	(NA)	(NA)	Non-modifiable device preconfigured to support serial connection between development host and target platform via AMIO console.

11.1.5 Modify User-Configurable Resources

Now that you know there are configurable resources within the target platform, the instructions in the following sections provide you with the opportunity to modify specific target platform resources.

Proceed to the following sections to learn more and to perform these procedures.

11.1.6 Assign Resources to a Target OS

Hypervisor comes with a default preconfiguration written into a *script file* named ***config.esh***. However, you should not modify this file directly. Instead, a script files called **override.esh** is used to modify the default settings. Use Workbench to view and modify **override.esh** contents to achieve the configuration you need.

Before modifying override.esh, make a backup copy. Should modifications to the script cause the target system to stop working correctly, you will be able to restore the backup copy and return to a working state.

To do this from Workbench, right-click on override.esh and select Copy. Right-click again and select Paste. In the dialog box, name the file override.bak

Make certain the development host and target platform are connected via serial and are powered up.

On the development host, open Workbench.

In Project Explorer, click the Hypervisor integration project icon (**wr_hypervisor_integration**) to expand the Hypervisor project contents. See Figure 168 - Hypervisor Project.

Figure 168 - Hypervisor Project

Project Explorer
common_pc (Wind River Linux Platform 4.3 Platform Project)
vip_wrVbX86_1 (Wind River VxWorks 6.9 Image Project)
vip_wrVbX86_2 (Wind River VxWorks 6.9 Image Project)
vsb_wrVbX86 (Wind River VxWorks 6.9 Source Build (Kernel Library) Project)
wr_hypervisor_integration (Host OS (Native Development) User-Defined Project)

Scroll down to the **config** folder and expand it by clicking its icon.

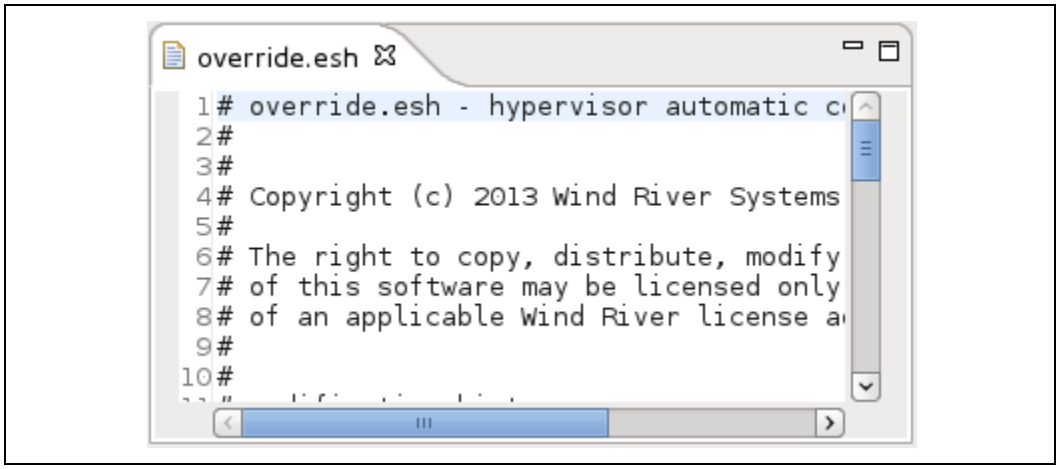
Under the **config** folder, several **ESH** files appear.

CAUTION: Unless specifically instructed in this document, do not open and/or modify the ESH files within the config folder. Doing so may cause the target platform to malfunction or stop operation.

11.1.6.1 OVERRIDE.ESH Script File

Double-click the **override.esh** script file. To the right of Project Explorer, an **override.esh** work pane appears. See Figure 169 - OVERRIDE.ESH Work Pane.

Figure 169 - OVERRIDE.ESH Work Pane

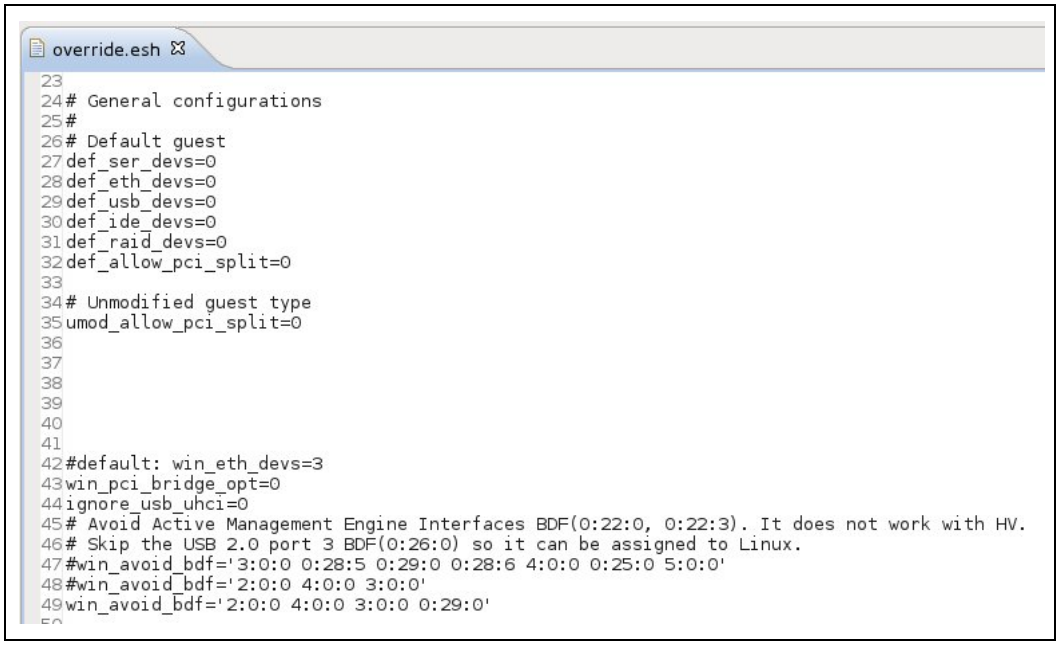


Double-click the **override.esh** work pane. The work pane expands.

Scroll up and down within the **override.esh** work pane and examine its contents.

Within the **override.esh** work pane are several sections that contain default target configuration values. See Figure 170 - OVERRIDE.ESH Configurable Section.

Figure 170 - OVERRIDE.ESH Configurable Section



In the configurable section in order to assign resources to specific OSes, later in this document you will learn how to modify the values inside **override.esh**'s configurable section.

Note that device assignment is done on an OS type basis. Thus, when specifying rules for VxWorks, ALL VxWorks instances will follow the same rules. If there is only a single device matching the assignment but multiple guest OS, then only the first guest OS will have the device. Other guest OS of the same type would not get the device.

Before configuring device assignment, first learn about device assignment rules, which are called *parameters*.

11.1.6.2 Device Assignment Parameters

When assigning devices to an OS, Hypervisor works in conjunction with **override.esh** scripting according to five general parameters:

11.1.6.2.1 Parameter 1: Assignment by default

If you do not assign device types to an OS, then the Hypervisor assigns devices according to a default configuration in **override.esh**.

This, unless otherwise specified in the specific OS parameters, by default the system assigns a guest OS with:

- 1 serial port
- 1 Ethernet port

You can change the default device assignment, but there is normally no need to do so.

NOTE: You will have the opportunity to create customized settings later in this work flow.

11.1.6.2.2 Parameter 2: OS specific device assignment

While the default parameters specify assignment for Guest OSes in general, the hypervisor also supports OS-specific assignments. Table 7 - Supported Guest OSes shows the supported Guest OSes.

Table 7 - Supported Guest OSes

Guest OS	prefix	Description
VxWorks	vx	Hypervisor-aware VxWorks OS
Linux	lx	Hypervisor-aware Wind River Linux

In our case, as our Guest OSes consist of two VxWorks instances and a single unmodified Wind River Linux instance, we use the vx and umod prefixes. The other OS assignments will not have an effect on our system.

Since each OS is assigned to a unique core, they will boot simultaneously. However, when the script assigns the devices, the order in which the OSes are defined control the assignment order.

1. **Guest OS 1:** Wind River Linux 5.0
2. **Guest OS 2:** Wind River VxWorks 1
3. **Guest OS 3:** Wind River VxWorks 2

Thus, the Linux GOS will get the first device in a given class, then VxWorks 1 will get the next device, etc…

It is important to realize that device assignment can only be controlled based on the OS type. As an illustration, there is no way to assign a specific device to VxWorks 2 and not VxWorks 1.

Parameter 3: Assignment by Availability

If a connected device has not already been assigned to another OS, then the system assigns it to the next OS defined – assuming that OS allows the device.

For example, assume the target platform hardware has only three USB controllers. Also assume you modify **override.esh** so that two USB controllers go to Guest OS 1, one USB controller to Guest OS 2 and one to Guest OS 3. The following (shown in Table 8 - Sample Scenario) happens as Hypervisor starts up.

Table 8 - Sample Scenario

Guest OS	# of USB Controllers Requested (of three total)	# of USB Controllers Actually Assigned
1	2	2
2	1	1
3	1	0

Despite configuring it to receive a USB controller, Guest OS 3 did not receive any USB devices because:

- A customized assignment in **override.esh** overrides the default USB assignment defined in Parameter 1 (No USB controllers are assigned by default).
- The previous two Guest OSes already loaded their assigned USBs (see Parameter 2).
- By the time Guest OS 3 begins its assignment, the number of available USBs defined in Parameter 3 are already depleted.

11.1.6.2.3 Parameter 4: "Avoid" Device Assignment

While the previous three parameters affect device assignment by addressing the *general type of device*, Parameter 4 addresses *individual device assignment*. When using Hypervisor and **override.esh,** it is not possible to assign a device directly to an OS type. However, you may use the **avoid** value to assign a device indirectly.

To assign a specific device to a type of Guest OS, use the **avoid** value.

Upon startup, by configuring Hypervisor to "avoid" loading a device to some OS types, you can force the device to load to the other OS.

There are two ways to use **avoid**:

- avoid [VendorID]:[DeviceID]
- avoid [bus]:[device]:[function]

Read on to learn more about these two forms of **avoid**.

avoid [VendorID]:[DeviceID]

By using this **avoid** statement, you stop a certain OS types from loading a specific device, and thus may allow another OS type to load it. Use this parameter when you know the vendor ID and device ID for a specific device.

For example, assume the following avoid line:

```
vx_avoid_ser='0x163C:0xFF02'
```

Where:

- **vx** means this rule applies to both VxWorks OSes.
- **avoid** means to avoid assigning this device to the indicated OS (VxWorks).
- **ser** means this rule applies to the device type (serial).
- **0x163c** signifies the vendor (Intel).
- **0xFF02** signifies the specific device ID.

To identify a device by its [VendorID]:[DeviceID] values, continue reading the next section (avoid [bus]:[device]:[function] – (BDF)). Near the end of that section are instructions for identifying a device by its [VendorID]:[DeviceID] value.

avoid [bus]:[device]:[function] – (BDF)

By using this **avoid** statement, you prohibit a certain OS type from loading a specific device, and thus may allow another OS type to load it. Use this parameter when you know the bus/device/function (BDF) value for a specific device.

For example, assume the following avoid line:

```
vx_avoid_ser='0:22:0'
```

Where:

- **vx** means this rule applies to both VxWorks OSes.

- **avoid** means to avoid assigning this device to the indicated OS (VxWorks).
- **ser** means this rule applies to the device type (serial).
- **0:22:0** signifies the specific device ID.

To determine a device's bus/device/function (BDF) when using **avoid [bus]:[device]:[function]**, refer to the Hypervisor bootup log.

Determine a Device's BDF

Follow this procedure to determine a device's BDF value:

In Project Explorer, click the Hypervisor integration project (**wr_hypervisor_integration**) icon to expand the Hypervisor project contents.

Scroll down to the **config** folder and expand it by clicking its icon.

Double-click the **override.esh** script file. To the right of Project Explorer, an **override.esh** work pane appears.

Scroll to the bottom of the **override.esh** work pane.

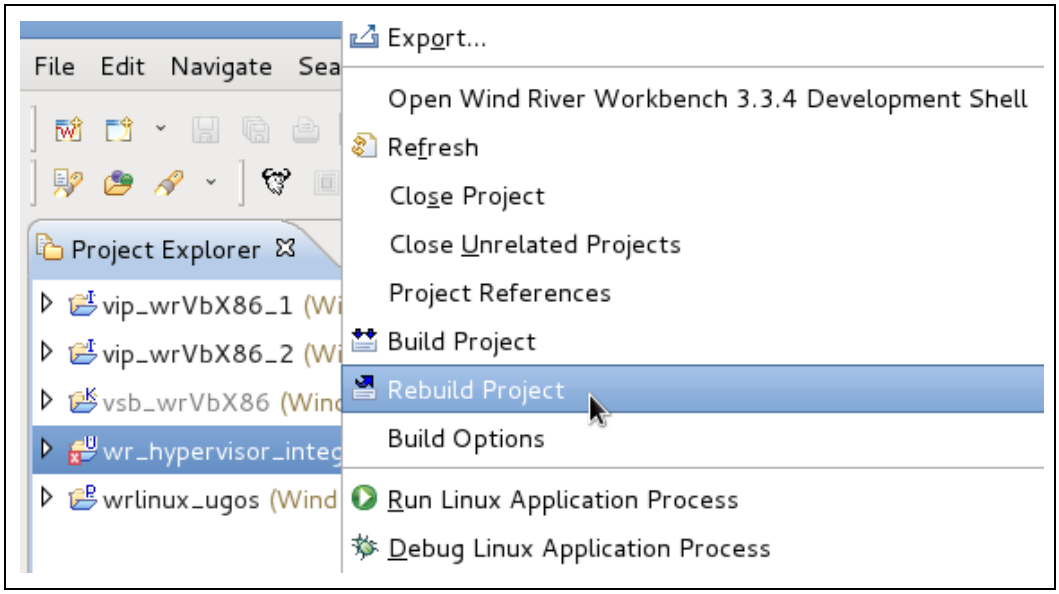
At the bottom of the work pane, type:

```
verbose=1
```

Go to **File > Save**. The system saves the change you made to the **override.esh** file.

Right-click the **wr_hypervisor_integration** object and select **Rebuild**. See Figure 171 - Rebuild Hypervisor Project.

Figure 171 - Rebuild Hypervisor Project



The system rebuilds the Hypervisor and your updated target platform configuration to a new **system.elf** file.

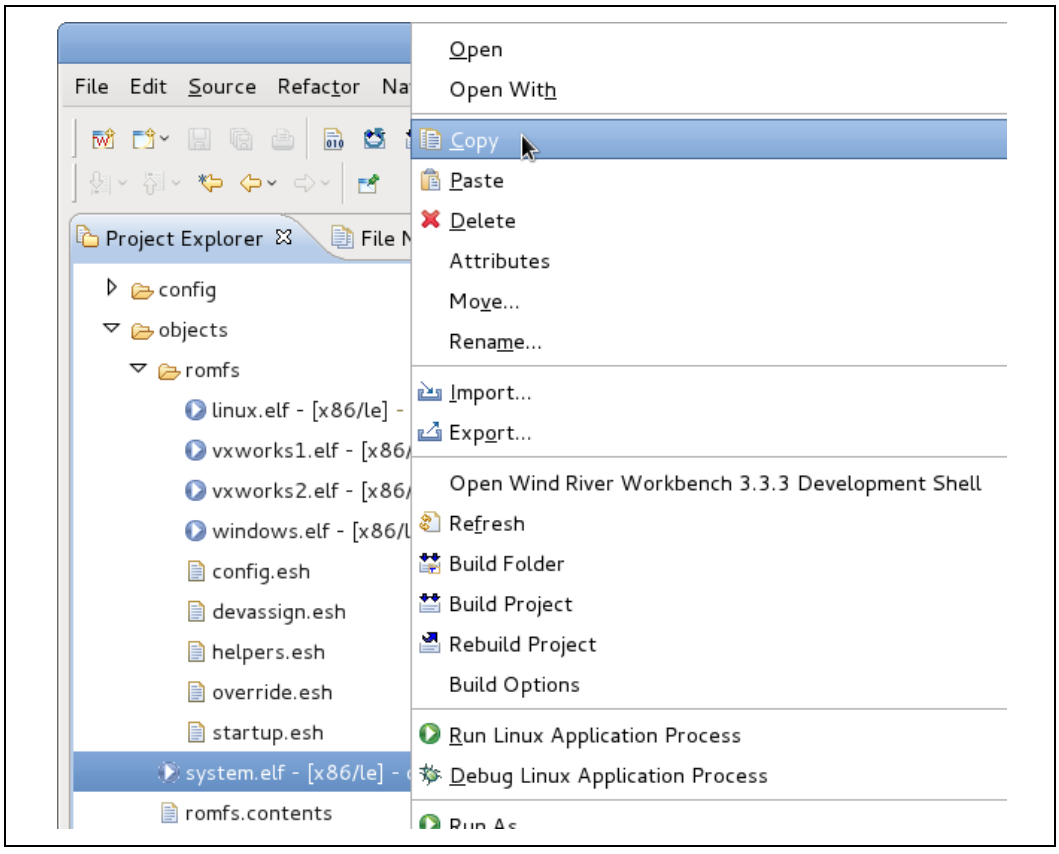
CAUTION: Intel Corporation provides you with the target platform media USB for copying the system.elf file from the development host to the target platform. However, copying data onto the target platform USB media may cause the data on the media to be overwritten. Make certain to back up data before using the media. Failure to do so may result in the loss of the data on the media.

At the development host, insert a Linux-formatted USB media (with 16 GB or more free space) into an open USB port. The USB media mounts to the Linux file system.

In Workbench Project Explorer, navigate through the **Objects > romfs** folders, and right-click the **system.elf** object.

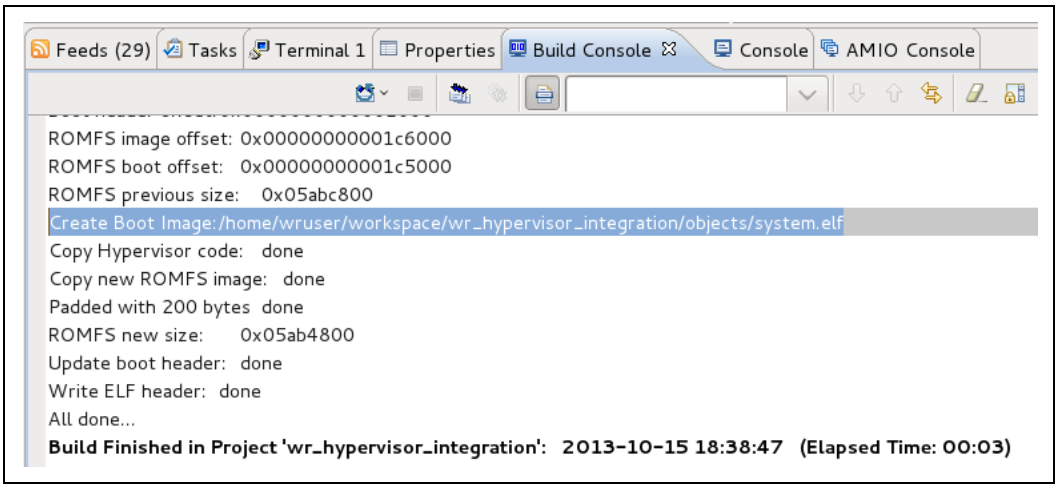
Select **Copy**. The **system.elf** file copies to the system. See Figure 172 – System.elf Copy.

Figure 172 – System.elf Copy



You may optionally use a command line or the Linux file explorer to copy the **system.elf** file. See the darkened content in Figure 173 – System.elf Directory.

Figure 173 – System.elf Directory



Minimize Workbench.

Use Linux File Explorer to navigate to the root directory of the USB media.

Paste the **system.elf** file to the root directory of the USB media.

After the file finishes copying, unmount the USB media and remove it from the development host.

Make certain that the target platform is powered down.

Ensure that no other bootable media are inserted into the target platform.

Insert the USB media into an open port on the target platform.

NOTE: For a faster boot, Intel Corporation recommends you connect the boot media to a USB 3.0 port on the target platform. Arrows in Figure 174 - USB 3.0 Ports provide the locations of two target platform USB 3.0 ports.

Figure 174 - USB 3.0 Ports

11.2 Configure Target Platform BIOS to Boot from USB Media

Before booting the target platform from USB media, you first configure the target platform BIOS to boot from USB media.

NOTE: To ensure that you start the target platform from USB media, follow this procedure each time you need to boot from USB media.

Shut down the target platform and wait 5 seconds.

Place the bootable USB media into an open USB port (preferably a blue USB 3.0 port).

Press and release the target platform power button. The target platform begins to start.

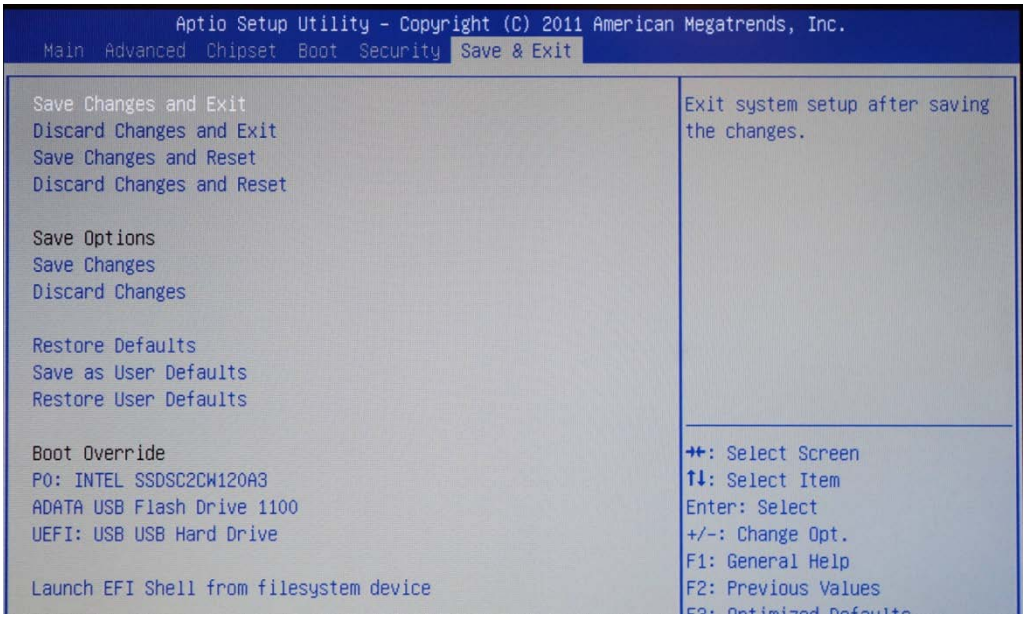
As the target platform begins to start, repeatedly press the **DELETE** button.

The target platform enters the BIOS **Main** tab. See Figure 175 – BIOS 1.

Figure 175 – BIOS 1

Use the right-arrow key to go to the **Save & Exit** tab. See Figure 176 – BIOS 2.

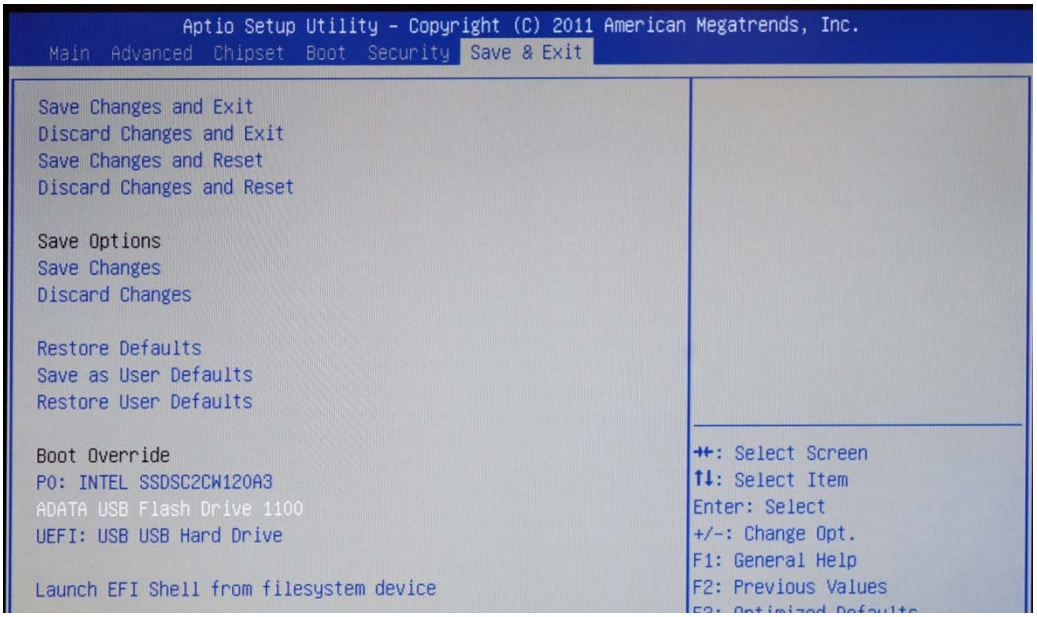
Figure 176 – BIOS 2



Use the down-arrow key to move the highlighted selection to the USB media. See Figure 177 – BIOS 3.

NOTE: In this example, the USB media is named **ADATA USB Flash Drive 1100**. Your USB media may have a different name.

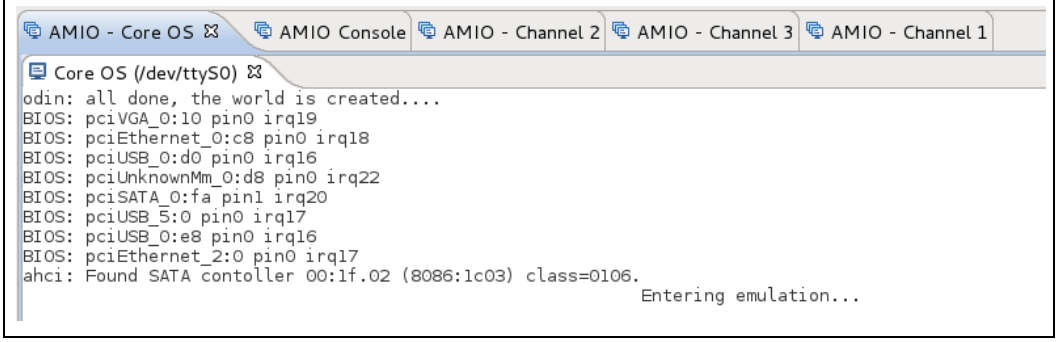
Figure 177 – BIOS 3



Press **Enter**. The target platform boots from USB media.

On the development host, connections to the guest OSes appear in individual AMIO consoles at the bottom of the development host Workbench workspace. Among these consoles appears the **AMIO Core** tab, which shows the contents of the Hypervisor startup log. See Figure 178 – Hypervisor AMIO Console.

Figure 178 – Hypervisor AMIO Console


AMIO - Core OS
AMIO Console
AMIO - Channel 2
AMIO - Channel 3
AMIO - Channel 1
Core OS (/dev/ttyS0)
odin: all done, the world is created....
BIOS: pciVGA_0:10 pin0 irq19
BIOS: pciEthernet_0:c8 pin0 irq18
BIOS: pciUSB_0:d0 pin0 irq16
BIOS: pciUnknownMm_0:d8 pin0 irq22
BIOS: pciSATA_0:fa pin1 irq20
BIOS: pciUSB_5:0 pin0 irq17
BIOS: pciUSB_0:e8 pin0 irq16
BIOS: pciEthernet_2:0 pin0 irq17
ahci: Found SATA contoller 00:1f.02 (8086:1c03) class=0106.
Entering emulation...

Allow the target platform to fully boot.

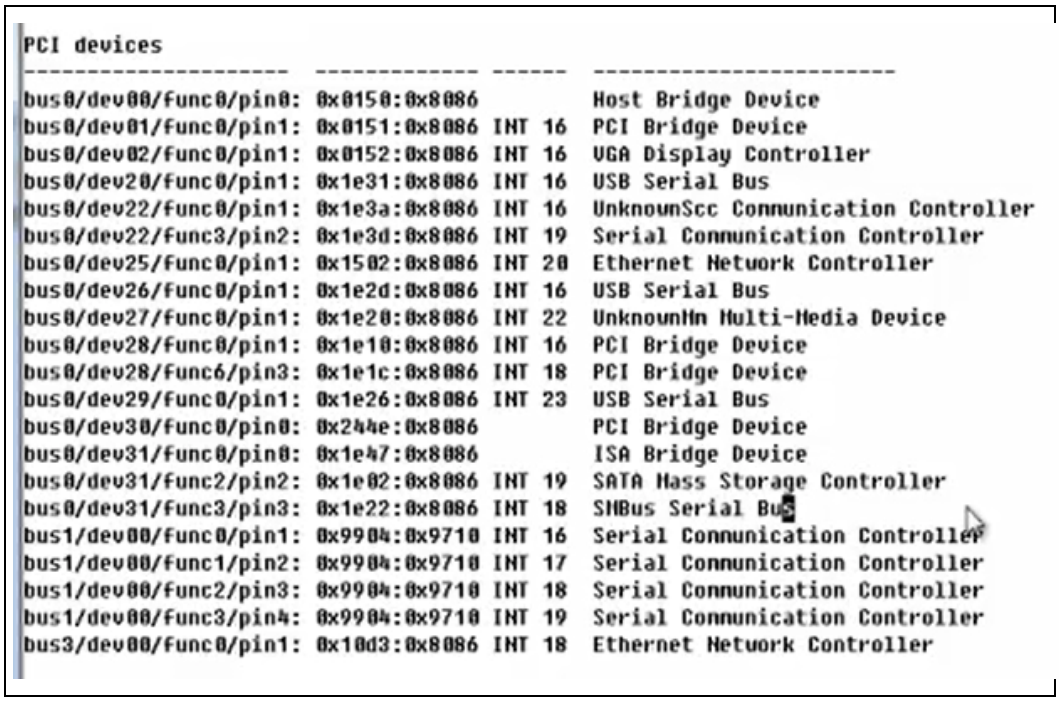
Scroll up and down within the Hypervisor console and examine its contents.

Within the Hypervisor AMIO console are several sections that contain boot information that include the devices recognized by Hypervisor.

Find the “PCI Devices” section of the Hypervisor console. See Figure 179 – Device Information.

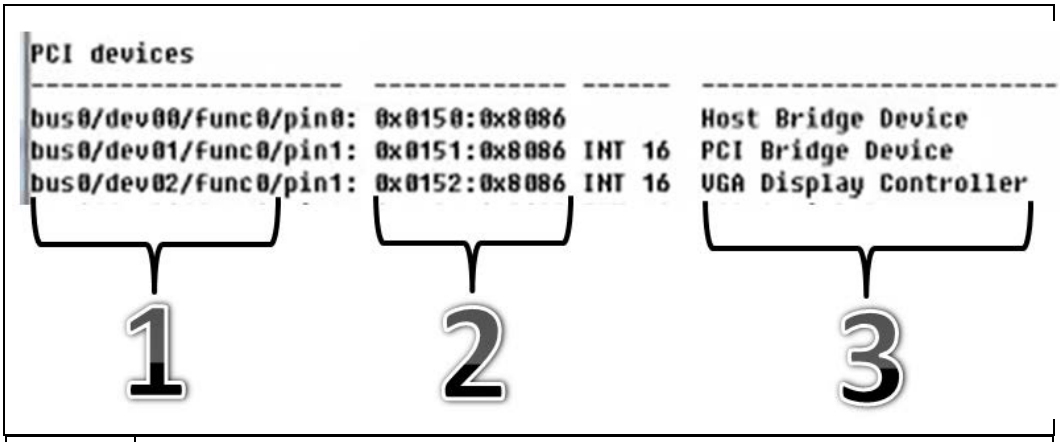
NOTE: This section appears approximately halfway down the Hypervisor startup log.

Figure 179 – Device Information


PCI devices
bus0/dev00/func0/pin0: 0x0150:0x8086 Host Bridge Device
bus0/dev01/func0/pin1: 0x0151:0x8086 INT 16 PCI Bridge Device
bus0/dev02/func0/pin1: 0x0152:0x8086 INT 16 VGA Display Controller
bus0/dev20/func0/pin1: 0x1e31:0x8086 INT 16 USB Serial Bus
bus0/dev22/func0/pin1: 0x1e3a:0x8086 INT 16 UnknownScc Communication Controller
bus0/dev22/func3/pin2: 0x1e3d:0x8086 INT 19 Serial Communication Controller
bus0/dev25/func0/pin1: 0x1502:0x8086 INT 20 Ethernet Network Controller
bus0/dev26/func0/pin1: 0x1e2d:0x8086 INT 16 USB Serial Bus
bus0/dev27/func0/pin1: 0x1e20:0x8086 INT 22 UnknownHm Multi-Media Device
bus0/dev28/func0/pin1: 0x1e10:0x8086 INT 16 PCI Bridge Device
bus0/dev28/func6/pin3: 0x1e1c:0x8086 INT 18 PCI Bridge Device
bus0/dev29/func0/pin1: 0x1e26:0x8086 INT 23 USB Serial Bus
bus0/dev30/func0/pin0: 0x244e:0x8086 PCI Bridge Device
bus0/dev31/func0/pin0: 0x1e47:0x8086 ISA Bridge Device
bus0/dev31/func2/pin2: 0x1e02:0x8086 INT 19 SATA Mass Storage Controller
bus0/dev31/func3/pin3: 0x1e22:0x8086 INT 18 SMBus Serial Bus
bus1/dev00/func0/pin1: 0x9904:0x9710 INT 16 Serial Communication Controller
bus1/dev00/func1/pin2: 0x9904:0x9710 INT 17 Serial Communication Controller
bus1/dev00/func2/pin3: 0x9904:0x9710 INT 18 Serial Communication Controller
bus1/dev00/func3/pin4: 0x9904:0x9710 INT 19 Serial Communication Controller
bus3/dev00/func0/pin1: 0x10d3:0x8086 INT 18 Ethernet Network Controller

To determine a specific device's BDF ID and [VendorID]:[DeviceID], refer to the values that appear in each line. See Figure 180 – Device BDF and [VendorID]:[DeviceID] Information.

Figure 180 – Device BDF and [VendorID]:[DeviceID] Information



Item	Comment
1	Device bus/device/function (BDF).
2	Device [VendorID]:[DeviceID].
3	Information about the kind of device.

§

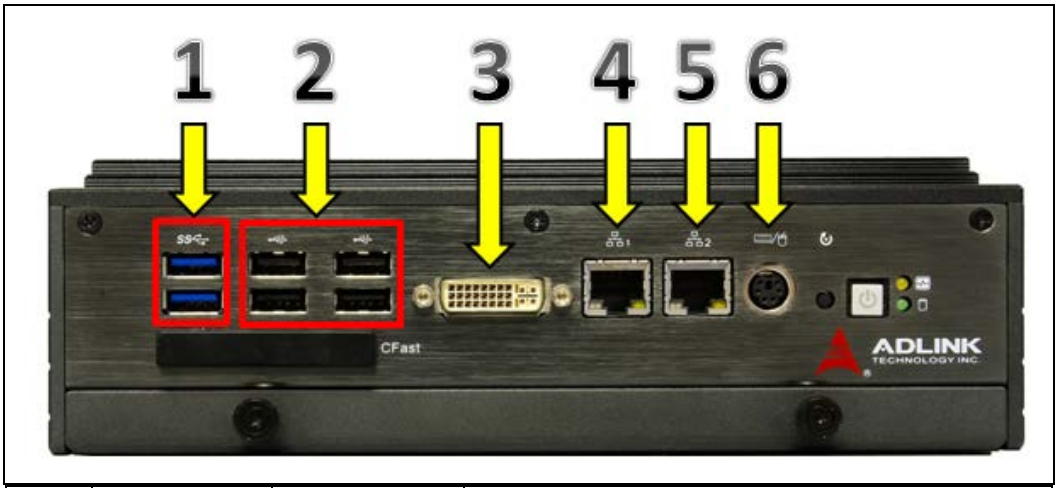
Appendix A – Default I/O Device Assignment

A.1 Default Physical Device Allocations

Default Physical Device Allocations

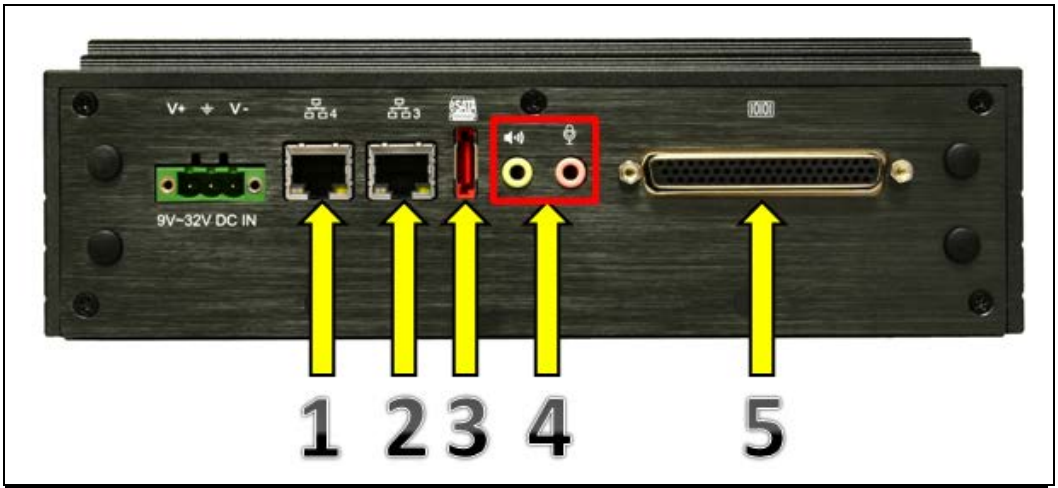
Intel® Industrial Solutions System Consolidation Series SCS comes with preconfigured physical device allocations as shown in Figure 181 – Target Platform Devices 1 and Figure 182 - Target Platform Devices 2.

Figure 181 – Target Platform Devices 1



Item #	Device Type	Assigned to VM	Comment
1	USB 3.0	Linux	Denoted by blue connector color.
2	USB 2.0	VxWorks 1	Denoted by black connector color
3	Video	Linux	Basic (non-graphical) command-line style video only supported at this time.
4	Ethernet	VxWorks 1	This device is driven by an Intel driver.
5	Ethernet	Linux	This device is driven by an Intel driver.
6	PS2	(NA)	Not supported by this implementation of SCS.

Figure 182 - Target Platform Devices 2



Item #	Device Type	Assigned to VM	Comment
1	Ethernet	VxWorks 2	This device is driven by a Realtek driver.
2	Ethernet	Linux	This device is driven by a Realtek driver.
3	eSATA	(NA)	Not supported by this implementation of SCS.
4	Audio	(NA)	Not supported by this implementation of SCS.
5	Serial	(NA)	Non-modifiable device preconfigured to support serial connection between development host and target platform via AMIO console.

§

Appendix B – Technical Support

B.1.1 In-Product Technical Support

In the event you require support while using Intel® Industrial Solutions System Consolidation Series (SCS), we strongly recommend you first consult with the many resources provided to you as documentation in printed and electronic file format in the product packaging.

- For a document listing, go to Section 3 – Intel® Industrial Solutions System Consolidation Series Documentation.
- For self-help activities, see Appendix C – Troubleshooting.

B.2 Online & Live Technical Support

As an authorized Intel® Industrial Solutions System Consolidation Series (SCS) product owner, you are entitled to technical support as defined on the SCS product website. See the technical support statement at:

www.intel.com/industrialconsolidation

B.3 Wind River Product Support & Training

Portions of your Intel® Industrial Solutions System Consolidation Series (SCS) product are provided courtesy of Wind River. For support and training beyond that defined here, see Wind River's website:

www.windriver.com/company/contact

§

Appendix C – Troubleshooting

CAUTION: Content within this section may include activities that entail risk to the preservation of data you may have accumulated in the course of your business activities. For example, in order to resolve a performance issue in your system, you may decide to follow the recommendation to reinstall all or part of the system software. In so doing, you may cause data that you may have accumulated in your system hard drive storage to be irretrievably formatted. To avoid data loss, make certain to create full backups of your business data onto remote resources in accordance with your company's policies. Make certain to review other Safety content outlined in Section 2 – Safety Notice.

This section provides activities you may perform to resolve some issues while using the Intel® Industrial Solutions System Consolidation Series (SCS).

This chapter is arranged into subsections when issues commonly occur:

- **Installation –** Issues that may arise when setting up, connecting, and starting the system
- **Configuration –** Issues that may arise when developing target platform configurations for production

Review the items listed in the sections below for a description of your issue and perform the suggested troubleshooting activity to help resolve the issue. Also make certain to consult the references mentioned in Appendix B – Technical Support.

IMPORTANT: This content may be updated without notice. To ensure that you are reading the most recent document, check the product support website:

www.intel.com/industrialconsolidation

In the event that all immediately available troubleshooting and help does not resolve your issue, interactive support is available. For more information, go to www.intel.com/industrialconsolidation

IMPORTANT: Some Recommend Resolutions shown below instruct you to perform activities on the development host, and some on the target platform. Make certain that you distinguish between these two subsystems, and that you perform activities on the subsystem indicated.

C.1 Installation Troubleshooting

The following troubleshooting items are commonly related to system installation.

Table 9 - Installation Troubleshooting

Item #	Issue	Recommended Resolution
T0001	Development host **does not boot** from the **USB** media labeled *Development Host*.	Turn off the development host. Make certain the *Development Host* USB media is plugged into a functional USB port on the development host. Start the development host. As the development host starts up, invoke the development host BIOS menu. Configure the BIOS so that the development host boots from USB media upon next startup. Boot the development host from the *Development Host* USB media. If the development host still does not boot from the *Development Host* USB media, try using a different USB port, or try using different development host hardware.
T0002	When attempting to **install development host software** to the development host, the **"Install to Hard Drive"** icon does not appear.	Before you can install the development host software onto the development host, you must first license the *Development Host* USB media. The licensing process includes placement of a license file into a specific file directory. Make certain you place the file into the correct location and follow the instructions provided in Section 6.6 – License the System. After placing the license file into the proper *Development Host* USB media directory and booting the development host from the USB media, it may take several minutes for the "Install to Hard Drive" icon to appear.

Item #	Issue	Recommended Resolution
T0003	When attempting to **install development host software** to the development host, the **installation takes too long**.	Installation time depends greatly upon the performance factors of the user-provided development host. These factors include processor speed, hard drive speed, and overall throughput among other factors. During testing on various development host hardware, Intel Corporation observed installation times ranging from 10 minutes to 2 hours. Generally, newer hardware systems require less installation time than older systems. Because software ideally only needs to be installed once, Intel Corporation recommends waiting until the system installs completely.
T0004	When attempting to **install development host software** onto the development host, the **installation hangs** or **fails**.	Make certain your development host hardware meets the minimum performance specifications defined in Section 4.1.2 – Development Host Specifications. If you determine you must install your development host software onto different hardware, you should first contact your Intel representative to nullify the old license and provide Host ID information for the new development host. See the licensing instructions in Section 6.6 – License the System.
T0005	When establishing an **AMIO connection** between the development host and target platform over **serial connection**, the development host and target platform **do not connect**.	Make certain you correctly set up the physical and logical connections between the development host and target platform. When starting up the development host and target platform and establishing a connection, make certain to start up the development host software first and establish an open connection on the development host AMIO console before starting the target platform. See Section 6 – Workflow 1: Set Up & Explore the System Features.

C.2 Configuration Troubleshooting

The following troubleshooting items are commonly related to system configuration.

Table 10 - Installation Troubleshooting

Item #	Issue	Recommended Resolution
T0006	When attempting to **boot the target platform** using the ***Target Platform* USB media**, the target platform **boots from the target platform hard drive** instead.	Every time you boot the target platform from USB media, you must configure the target platform BIOS settings to boot from USB media. After configuring the BIOS to boot from USB Media, this change is not persistent, and the system boots only *one time* from USB Media. On subsequent startups, the target platform boots from hard drive.
T0007	When attempting to **boot the target platform** using the ***Target Platform* USB media**, the **target platform hangs**.	Make certain the boot image file name is named **system.elf**. If the boot image has another name, rename it to **system.elf** and retry booting. The boot image on the *Target Platform* USB media may be corrupt. Recreate the image and try rebooting the target platform from it.
T0008	I load **customized code into my VxWorks / Linux project** and create a **new system.elf that I copy to USB**. After booting the target platform using the USB media with the image containing my customized code, the **code does not appear in the VxWorks / Linux OSes after the target platform boots**.	If the customized code doesn't appear upon booting up the target platform, the target platform likely did not boot from the USB media. Every time you boot the target platform from USB media, you must configure the target platform BIOS settings to boot from USB media. After configuring the BIOS to boot from USB Media, this change is not persistent, and the system boots only *one time* from USB Media. On subsequent startups, the target platform boots from hard drive.

Item #	Issue	Recommended Resolution
T0009	While using the development host software to **build a new Linux project from scratch**, the **build process takes too long**.	When frequently adding small portions of customized code or small components that require frequent recompiling of the Linux project, try using the **Rebuild** command instead the **Build** command. Rebuild recompiles the Linux project, adding only the differences you added since the last full project build. Using Rebuild can significantly reduce overall wait time as the project compiles. To further reduce build time, select fewer components and / or work with a development host with powerful processing capabilities. During testing on various development host hardware, Intel Corporation observed Linux project build times ranging from 15 minutes to 8 hours.
T0010	When **booting the target platform** from USB media containing a **system.elf** file I created using the development host software, not all **OSes boot up on the target platform**.	Firstb confirm that you are actually booting the target platform from the USB media. See Troubleshooting item #T0006. Allow the target platform to restart form the USB boot media containing the **system.elf** file you wish to boot. In the development host AMIO pane, consoles for each OS should open. If all of the OS consoles do not appear within a few minutes after booting the target OS, then not all of the OS projects you created in the development host were integrated properly into the bootable **system.elf** file. Make certain you have followed the instructions in the *User Guide* chapters entitled *Develop With Preloaded Workbench Projects* or *Build Linux & VxWorks OS Images from Scratch*. These chapters provide instructions to ensure that all OS projects are properly compiled and integrated into a bootable **system.elf** file.

§

Appendix D – Legal Notices

D.1 All Notices

Make certain to read all information including legal notices before using the system. Information is available in printed and electronic formats from:

- this document, the *Intel® Industrial Solutions System Consolidation Series (SCS) User Guide*
- the *User Support* media USB
- the development host software system. See Section 3 – Intel® Industrial Solutions System Consolidation Series Documentation.
- Reference materials included those listed in Section 3 – Intel® Industrial Solutions System Consolidation Series Documentation.
- www.Intel.com
- www.WindRiver.com
- your sales and support representatives

D.2 License Notices

Access an important *Wind River Linux 5.0 Third Party License Notices* document by going to the following relative path on the development host media:

…/home/wruser/WindRiver/legal-notices/wr-Linux-5.0.1/
WindRiver_Linux5.0_ThirdPartyNotices_v2.1.pdf

Access other license documents by going to the following relative path on the development host media:

…/home/wruser/WindRiver/licenses

IMPORTANT: Make certain to read important information regarding GNU General Public License version 3 (GPLv3) and GNU General Public License version 2 (GPLv2) in the *Wind River Linux User's Guide*, page 112. Access this document as an Adobe PDF file by going to the following relative path on the Development Host:

…/home/wruser/WindRiver/docs/extensions/eclipse/plugins/com.windriver
.ide.doc.wr_linux_5/wr_linux_users_guide/
wr_linux_users_guide_5.0.1.pdf

D.3 GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this

version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty

adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified

versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require

acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or

other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail. If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>

§